WEST

End of Result Set

☐ Generate Collection | Print

L7: Entry 2 of 2 File: USPT Oct 6, 1998

US-PAT-NO: 5818438
DOCUMENT-IDENTIFIER: US 5818438 A

TITLE: System and method for providing television services

DATE-ISSUED: October 6, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Howe; Wayne R.	Dublin	OH		
Danner, III; Fred Thomas	Alpharetta	GA		
Mauney; John R.	Jefferson	GA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
BellSouth Corporation	Atlanta	GA			02

APPL-NO: 08/ 428718 [PALM]
DATE FILED: April 25, 1995

INT-CL: [06] H04 N 7/173

US-CL-ISSUED: 345/327; 348/13, 455/4.2
US-CL-CURRENT: 345/718; 345/721

FIELD-OF-SEARCH: 348/9, 348/6, 348/10, 348/12, 348/13, 348/14, 348/15, 348/16, 348/17, 348/18, 348/461, 348/463, 348/465, 348/468, 348/473, 348/478, 348/553, 348/563, 348/564, 348/565, 348/566, 455/5.1, 455/6.1, 455/6.2, 455/6.3, 455/4.2, 455/3.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	3798610	March 1974	Bliss et al.	
☐	3886302	May 1975	Kosco	
☐	4130833	December 1978	Chomet	
☐	4258386	March 1981	Cheung	
☐	4361851	November 1982	Asip et al.	
☐	4488179	December 1984	Kruger et al.	
☐	4566030	January 1986	Nickerson et al.	
☐	4567591	January 1986	Gray et al.	
☐	4598288	July 1986	Yarbrough et al.	
☐	4688248	August 1987	Tomizawa	
☐	4689661	August 1987	Barbieri et al.	
☐	4698670	October 1987	Matty	
☐	4720873	January 1988	Goodman et al.	
☐	4816904	March 1989	McKenna et al.	
☐	4890322	December 1989	Russell, Jr.	
☐	4912552	March 1990	Allison, III et al.	
☐	5010585	April 1991	Garcia	
☐	5046090	September 1991	Walker et al.	
☐	5046092	September 1991	Walker et al.	
☐	5055924	October 1991	Skutta	
☐	5173900	December 1992	Miller et al.	
☐	5191645	March 1993	Carlucci et al.	
☐	5208665	May 1993	McCalley et al.	
☐	5247347	September 1993	Litteral et al.	
☐	5249044	September 1993	Von Kohorn	
☐	5287181	February 1994	Holman	
☐	5335277	August 1994	Harvey et al.	
☐	5339315	August 1994	Maeda et al.	348/7
☐	5343240	August 1994	Yu	
☐	5357276	October 1994	Banker et al.	348/7
☐	5374951	December 1994	Welsh	
☐	5404393	April 1995	Remillard	
☐	5446490	August 1995	Blahut et al.	348/7
☐	5483277	January 1996	Granger	
☐	5502499	March 1996	Birch et al.	
☐	5585838	December 1996	Lawler et al.	348/906
☐	5608448	March 1997	Smoral et al.	348/7

ART-UNIT: 271

PRIMARY-EXAMINER: Grant; Chris

ABSTRACT:

A system and method are described for providing interactive television services and for switching between television programs, such as to an interactive program session from another program. An interactive server, capable of providing requested interactive video services to a set of subscribers in a given geographic area, is coupled to a network to which subscriber set top boxes are also coupled. Television programming, which may be furnished by any number of sources, is accompanied by a signal that indicates the availability of other programming, such as interactive television service related to the program being viewed, as well as information to be used in requesting such service. The user, when notified by this signal, may input to the set top box a request for a second program, such as an interactive program or application. The identity of the original program channel is stored, and a session with the video service provider is established over the network. When the session is terminated, the set top box re-tunes the television signal to the original program channel.

40 Claims, 17 Drawing figures

Citation

International Journal of Network Management >archive
Volume 7 , Issue 1 January–February 1997 >toc

Synchronization in multimedia data retrieval

Authors
Anna Haj Ha?
Cindy X. Xue

Publisher
John Wiley & Sons, Inc. New York, NY, USA

Pages: 33 - 62 Periodical-Issue-Article
Year of Publication: 1997
ISSN:1099-1190

doi>10.1002/(SICI)1099-1190(199701/02)7:1<33::AID-NEM232>3.0.CO;2-#

> full text > abstract > references > index terms > peer to peer

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format

↑ **FULL TEXT:** Access Rules

pdf 488 KB

↑ **ABSTRACT**

Synchronization of multiple medium streams in real time has been recognized as one of the most important requirements for multimedia applications based on broadband high-speed networks. This article presents a complete synchronization scheme for distributed multimedia information systems. © 1997 John Wiley & Sons, Ltd.

↑ **REFERENCES**

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 T. D. C. Little and A. Ghafoor, Synchronization and storage models for multimedia objects, IEEE Journal on Selected Areas in Communications, 8, No. 3, April, 413-27, 1990.

2 T. D. C. Little and A. Ghafoor, Network consider-ations for distributed multimedia object composition and communication, IEEE Network, Nov- ember, 32-49, 1990.

3 L. H. K. Pung, T. S. Chua and S. F. Chan, Temporal synchronization support for distributed multimedia information systems, Computer Communications, 17, No. 12, December, 852-62, 1994.

4 B. Furht, Multimedia systems: an overview, IEEE Multimedia, 1, No. 1, Spring, 47-50, 1994.

5 R. Steinmetz, Synchronization properties in multi- media systems, IEEE Journal on Selected Areas in Communications, 8, No. 3, April, 401-12, 1990.

6 T. D. C. Little and A Ghafoor, Multimedia synchron-efficient ization protocols for broadband integrated services, IEEE Journal on Selected Areas in Communications, 9, dupli-No. 9, December, 1368-81, 1991.

7 C. Nicolaou, An architecture for real-time multime- dia communication systems,. IEEE Journal on Selected Areas in Communications, 8, No. 3, April, 391-400, 1990.

8 W-H. F. Leung, T. J. Baumgartner and Y. H. Hwang, A software architecture for workstations supporting multimedia conferencing in packet switching net- works, IEEE Journal on Selected Areas in Communi- cations, 8, No. 3, April, 380-89, 1990.

9 Domenico Ferrari, Delay jitter control scheme for packet-switching internetworks, Computer Communications, v.15 n.6, p.367-373, July/Aug. 1992

10 D. Ferrari, Design and applications of a delay jitter control scheme for packet-switching internetworks, Proc. of the 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video, 1991, Lecture Notes in Computer Science, Springer-queue Verlag, Berlin, Vol. 614, 1992, pp. 72-83.

11 V. Rangan, S. Ramanathan, H. M. Vin and T. Kaeppner, Techniques for multimedia synchroniz- ation in network file systems, Computer Communi- cations, 16, No. 3, March, 168-76, 1993.

12 P. Venkat Rangan , Srinivas Ramanathan , Thomas Kaeppner, Performance of inter-media synchronization in distributed and heterogeneous multimedia systems, Computer Networks and ISDN Systems, v.27 n.4, p.549-565, Jan. 1995

13 T. D. C. Little , A. Ghafoor, Scheduling of bandwidth-constrained multimedia traffic, Computer Communications, v.15 n.6, p.381-387, July/Aug. 1992

14 Lamont and N. D. Georganas, Synchronization architecture and protocols for a multimedia news service application, Proc. of the International Confer- ence on Multimedia Computing and Systems, pp. 3-8, 1994.

15 L. A. Karmouch and N. D. Georganas, Synchroniz-ation in real time multimedia data delivery, Proc. IEEE International Conference on Communications, Vol. 2, pp. 587-91, 1992.

16 S. H. Son and N. Agarwal, Synchronization of tem-poral constructs in distributed

multimedia systems with controlled accuracy, Proc. of the International Conference on Multimedia Computing and Systems, pp. 550-5, 1994.

17 Ravindran and V. Bansal, Delay compensation protocols for synchronization of multimedia data streams, IEEE Transactions on Knowledge and Data Engineering, 5, No. 4, August 574-89, 1993.

18 Doug Shepherd , Michael Salmony, Extending OSI to support synchronization required by multimedia applications, Computer Communications, v.13 n.7, p.399-406, Sep. 1990

19 M. Woo, N. U. Qazi and A. Ghafoor, A synchroniz-pp. ation framework for communication of pre-orches-30. trated multimedia information, IEEE Network, January/February, 52-61, 1994.

20 L. A. Karmouch and N. D. Georganas, Multimedia segment delivery scheme and its performance for real-time synchronization control, 1994 IEEE Inter-Towards national Conference on Communications, Vol. 3, pp. 1734-8, 1994.

21 N. B. Pronios and T. Bozios, A scheme for multime-72, dia and hypermedia synchronization, Proc. of Inter-32. national COST 237 Workshop on Multimedia Transport and Teleservices, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 340- 5, 1994.

22 David E. McDysan , Darren L. Spohn, ATM: theory and application, McGraw-Hill, Inc., New York, NY, 1994

23 D. Kohler and H. Muller, Multimedia playout synchronization using buffer level control, Proc. of the 2nd Int. Workshop on Multimedia: Advanced Tele- services and High-speed Communication Architectures, 1994, Lecture Notes in Computer Science, Springer- Verlag, Berlin, Vol. 868, pp. 167-80, 1994.

24 R. Steinmetz and C. Engler, Human perception of media synchronization, IBM European networking center, Technical report, 43.9310, 1993.

25 K. Nahrstedt and R. Steinmetz, Resource manage- ment in networked multimedia systems, Computer, 28, No. 5, May, 52-63, 1995.

26 L. Fedaoui, A. Seneviratne and E. Horlait, Implementation of an end-to-end Quality of Service Management scheme, Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices, 1994, Lecture Notes in Computer Science, Springer- Verlag, Berlin, Vol. 882, pp. 124-44, 1994.

27 H. Bowman, L. Blair, G. S. Blair and A. G. Chetwynd, A formal description technique support- ing expression of quality of service and media sych- ronization, Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices, 1994, Lec-ture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 145-67, 1994.

28 M. Diaz and P. Senac, Timed stream petri nets: a model for timed multimedia information, Proc. of 15th International Conference on Application and Theory of Petri Nets. 1994, Lecture Notes in ComputerScience, Springer-Verlag, Berlin, Vol. 815, pp. 219- 38, 1994.

29 D. R. C. A. Bulterman and R. Van Liere, Multimedia synchronization and UNIX, Proc. of the 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video, 1991, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 614, synchroniz-pp. 108-19, 1992. orches-

30 T. Znati and B. Field, A network level channel abstraction for multimedia communication in

real- time networks, IEEE Transactions on Knowledge and Data Engineering, 5, No. 4, August, 590-99, 1993.

31 B. L. Dairaine, L. Fedaoui, W. Tawbi and K. Thai, Inter-Towards an architecture for distributed multimedia applications support, Proc. of the International Confer- ence on Multimedia Computing and Systems, pp. 164- multime-72, 1994. Inter-

32 R. Steinmetz, Analyzing the multimedia operating system, IEEE Multimedia, 2, No. 1, Spring, 68-84, 1995.

33 D. L. Stone and K. Jeffay, Queue monitoring: a delay jitter management policy, 4th International Workshop on Network and Operating System Support for Digital Audio and Video, 1993, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 846, pp. 149- 60, 1994.

34 T. D. C. Little and F. Kao, An intermedia skew con- trol system for multimedia dat presentation, Proc. of the 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, 1992, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 712, pp. 130-41, 1993.

35 W. Yen and I. F. Akyildiz, On the synchronization mechansims for multimedia integrated services net- works, Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 168-84, 1994. [36] Z. Y. Shae, P. C. Chang and M. Chen, Capture and playback synchronization in video conferencing, Proc. on Multimedia Computing and Networking, 1995. SPIE, Vol. 2417, pp. 90-101, 1995. [37] G. J. Thaler, Automatic Control Systems, West Pub- lishing, St Paul, MN, 1989. 38. R. C. Dorf, Modern Control Systems, Addison-Wesley, Reading, MA, 1992.

↑ **INDEX TERMS**

Primary Classification:
C. Computer Systems Organization
↳ C.2 COMPUTER-COMMUNICATION NETWORKS

↑ **Peer to Peer - Readers of this Article have also read:**

Editorial pointers
Communications of the ACM 44, 9
Diane Crawford

News track
Communications of the ACM 44, 9
Robert Fox

Forum
Communications of the ACM 44, 9
Diane Crawford

New Products
Linux Journal 1996, 27es
CORPORATE Linux Journal Staff

Book Review: IPv6: The New Internet Protocol
Linux Journal 1996, 25es
CORPORATE Linux Journal Staff

INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT, VOL. 7, 33–62 (1997)

Synchronization in Multimedia Data Retrieval

Synchronization of multiple medium streams in real time has been recognized as one of the most important requirements for multimedia applications based on broadband high-speed networks. This article presents a complete synchronization scheme for distributed multimedia information systems. © 1997 by John Wiley & Sons, Ltd.

Int. J. Network Mgmt., vol. 7, 33–62(1997)
(No of Figures: 15. No of Tables: 16. No of Refs: 38)

By Anna Hać and Cindy X. Xue

Introduction

Multimedia refers to the integration of text, image, audio, and video in a variety of application environments. Recent developments in fiber optics and broadband communication technology such as B-ISDN and ATM provide bandwidth and delay characteristics necessary to support these new media services. A multimedia system can transmit either live video or audio as in teleconferencing or use stored video, audio, and image information as in a distributed multimedia information system (DMIS).[1–3] The DMIS system supports integration and coordination of audio, video, text, and image originating from different servers interconnected by high-speed networks. A server can be dedicated to the storage of either data of a single medium type of multiple media types. Information of multiple media types is retrieved by a client from servers and presented to the user in a meaningful fashion. New networks, protocols, and operating systems are necessary to meet the requirements of multimedia applications.

One of the requirements of a multimedia system is the need to provide synchronization of multiple media data elements which are sent via different channels and which experience random delays during the transmission period.[2–21] Traditional data communication networks are used to provide error-free transmission, and are not able to meet the real-time delivery requirements or to achieve synchronization. The task of synchronization is to eliminate all the variations and delays incurred during the transmission of the multiple media streams and to maintain synchronization among them.

End-to-end delay in distributed multimedia system consists of all the delays created at the source site, network, and receiver site. These delays usually consist of the packetization delay, the network access and transmission delay, the protocol pro-

Anna Hać and Cindy X. Xue teach in the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, Hawaii.

CCC 1055-7148/97/010033-30$17.50

One of the requirements of a multimedia system is the need to provide synchronization of multiple media data elements which are sent via different channels and which experience random delays during the transmission period.

cessing delay, and the presentation delay. Synchronization is much more complicated if media data involved in the same presentation originated from different sources in the network. Another source of asynchrony is clock drift, the mismatch between the sender's transmission rate and receiver's presentation rate. If the playout rate of the receiver is faster than the transmission rate of the sender, the receiver may suffer starvation. The receiver may suffer overflow if the receiver's transmission rate is slower than that of the sender.

There are two types of media data: discrete and continuous. Continuous media data, such as voice and video, have time-dependent values. They require real-time service delivery and fine-grain synchronization. An example is the synchronization of mouth movement of video sequences with audio presentation, which is also known as 'lip-sync'. This synchronization is more difficult to achieve when the video data and the corresponding audio data are sent from separate locations over a network. Discrete media data, in contrast, have time-independent values, such as text and graphics. They pose less stringent end-to-end delivery requirements yet need superior reliability in transmission. However, if this data has synthetic timing relationships with other data, as in the case of a slide show with blocks of audio allotted to each slide, it should be delivered on time without error. Based on these two types of media data, synchronization of multimedia includes both intra- and inter-media synchronization. While inter-media synchronization needs to maintain an inter-media relationship, intra-media synchronization guarantees that continuous media are played back at their fixed rates.

To achieve media synchronization, several synchronization requirements have to be identified and solved. First, it is desirable to have some useful tools or means to express complex temporal relationships of multiple media data in applications involving the simultaneous presentation of multiple media types. Second, real-time communication protocols and operating systems are essential. These sytems assign the utmost priority to the meeting of deadlines for a set of scheduled time-dependent tasks. Specific scheduling is required for storage devices, processors, and communication resources in order to overcome the latency of data, from its source to its destination. Third, efficient buffer management is needed to smooth variations in end-to-end delay for media units. End-to-end delay varies according to the transmission path lengths, network traffic load, and the host's CPU latency, which is also load dependent.

Research has been done in the media synchronization area. Steinmetz[5] presents a general concept of programming constructs for expressing time relationships among media. Little and Ghafoor[1,6] use Petri nets to define a synchronization model called OCPN for the description of presentation sequences and relationships of multimedia objects. Nicolaou[7] presents a method for implementing synchronization among related streams by inserting synchronization points in each individual stream. Leung[8] proposes a scheme using a multimedia virtual circuit. The transport layer of the sender multiplexes all the interrelated medium data into a single virtual circuit. Ferrari[9,10] presents a delay jitter control system that insures media synchronization as long as a bound on delay can be guaranteed and the sender and receiver clocks are kept in synchrony. Every node on the transmission path in the network is involved in synchronization control by regulating the random delay which has been introduced to an arriving packet by the previous hop. This method may be hard to implement in ATM networks because it introduces too much workload on the intermediate nodes that carry thousands of connections. The adaptive feedback method presented by Rangan[11,12] solves the asynchrony in multimedia on-demand services. It requires an additional connection for each sender and receiver pair to transmit feedback units. The centralized multimedia server uses feedback units transmitted by the receiver to detect the asynchronies among them. Correction of the asynchrony is made at the server by speeding up or slowing down the traffic using skip or pause media data on the various media streams. Adaptive feedback is not suitable for WANs because its synchroniz-

ation guarantees are limited by the maximum network delay. Little *et al.*[6,13] propose a scheduling protocol for stored media applications. They use an Object Composition Petri Net (OCPN) to capture the temporal relationship among media units. According to the OCPN and end-to-end delays, the playout schedule and transmission schedule for receiver and sender can be calculated respectively. Senders and receivers will then follow the computed schedules to transmit and play back the media data. However, due to the non-deterministic random delay of the network, scheduling and predicting the traffic is not sufficient to maintain a simultaneous data delivery. Also, the scheme does not consider the clock drift problem. Lamont and Georganas[14] present a stream synchronization protocol aiming to provide multimedia news services. The proposed protocol adds 'intentional delay' to those media streams which experience less end-to-end delay without violating the tolerant asynchrony bound specified by the application. The local clock drift problem is not considered in this article, nor is the way of correcting transmission rates of different servers. This method is difficult to apply to a WAN area where the network delay jitter is relatively large.

In this article we focus on distributed multimedia information systems (DMIS) in which data applications are stored. DMIS uses preorchestrated, stored multimedia data, and requires management of heterogeneous data with vastly different storage, communication, and presentation requirements. In DMIS, synchronization is a complex problem because several streams from independent sources can require synchronization with each other, in spite of the asynchronous nature of the network. We present a complete synchronization scheme for multiple data streams generated from distributed media database servers without using a global clock. Besides the using of OCPN model to describe the complex temporal relationship among coupled medium streams in applications, media scheduling at the server side and buffer reservation at the client side are needed during the connection, set-up phase. A buffer size configuration method is presented in reference 3, but it considers only intra-media synchronization. In order to achieve inter- as well as intra-media synchronization, determination of buffer size for one medium needs to consider the effect of other related media. The medium retrieval time plays an important role here. If the coupled media are retrieved at the same time, the buffer size of one medium has to be configured based on the maximum delay experienced by related media streams instead of its own medium delay. If they are retrieved at each corresponding control time, then there is no need to consider the delay effect of other media. Synchronization recovery is performed at the receiver end before the playback. To manage the buffer level behavior efficiently, the control methods are introduced. In reference 23, a scheme to control the average level of the receiver frame buffer to a nominal value is proposed to achieve the intra-media synchronization goal. Since the buffer is supposed to compensate delay jitter, the dynamic level change of buffer is normal. Instead of fixing the nominal value at one point in the buffer, we propose three different nominal values according to three different buffer states: underflow, normal, and overflow. Skew tolerance parameters[24] are used here as control constraints. In order to realize the intermedia synchronization goal, a master media-based control method is also provided.

Architecture of Distributed Multimedia Information Systems

An ideal network for multimedia communications is a constant-delay network, which could be characterized also as a zero-jitter network, where jitter (more precisely, delay jitter) is variation of stream packet delays. With this ideal network and real-time operating system available both at the server and client sides, synchronization of multimedia data becomes quite simple, because both the network transmission delay and operating system latency are bounded. However, this is usually not the case in practice, and it does not consider the clock drift problem.

The computers and the high-speed packet-switched networks that connect different servers and clients involved in multimedia communications may introduce large amounts of jitter because of fluctuation of their loads. In this article we assume that the broadband network interconnecting the multimedia servers with clients is a high-speed network, and on a wide area range. Therefore, the DMIS we study imposes non-deterministic delays in data retrieval due to query

evaluation, seek, and access delay at the server side; transmission delays due to network transmission, packetization, buffering, and depacketization; and presentation delays due to decompression of data at the client side.

—Quality of Service for Media Applications—

Implementing a synchronization algorithm for a specific multimedia application requires specifying the quality of service (QoS) for multimedia communications.[25,26] The QoS is a set of parameters that characterize communication services, including bandwidth, the bounds on tolerable end-to-end delays, jitter, and the level of reliability in transmission such as bit error rate and packet error rate.

QoS may be classified into three different parts: application QoS parameters, which stipulate the requirements for the application services; system QoS parameters, which describe the communication and operating system requirements resulting from the application QoS; and finally network QoS parameters, which give the network performance requirement such as bandwidth and jitter. Here we focus on the network QoS. With this network QoS specification available, the transport requirements for various types of media traffic are defined, and the network can allocate and reserve resources for multimedia applications.

Traditional communications protocols (like TCP/IP) are designed for data communications and are not suitable for the diversity of traffic and real-time requirements in multimedia communications. They are used to provide error-free service, while many multimedia services can tolerate errors in transmission due to packet corruption or loss without retransmission or correction. In fact, in order to meet the real-time delivery requirements, late packets are discarded to meet the deadlines of others. This asumption is based on the fact that some video and audio applications do not seriously degrade the service with dropped packets.

—Multimedia Communications in B-ISDN—

In a high-speed network like B-ISDN, multiple medium streams, such as text, image, voice, and video are first chopped into fixed-length packets, then multiplexed into a single bit stream which is transmitted across a physical medium. How to support multiple QoS classes for differing application requirements on delay and loss performance is a critical issue. Several methods[26,27] have been proposed to deal with the multimedia communication service. One proposed transport mechanism for B-ISDN is called asynchronous transfer mode (ATM).[22] ATM is a cell-based switching and multiplexing technology designed to be a general-purpose, connection-oriented transfer mode for a wide range of services. B-ISDN network, which is based on this technology, is emerging as the preferred means for multimedia communications.

An ATM network includes a physical layer, an asynchronous transfer mode (ATM) layer, an ATM adaptation layer (AAL) and higher layers. The basic transmission unit in an ATM network is the cell, which has a constant length of 53 bytes composed of a 5-octet header and a 48-octete payload. The header includes information about media access, connection and priority control of the cell. Each ATM cell sent into the network contains addressing information that establishes a virtual connection from source to destination. Virtual channels (VCs) are associated with cells by an identical virtual channel identifier (VCI) within their header, therefore no channel bandwidth is occupied in a virtual circuit except during real transmission of an ATM cell. And end-to-end path is established by the concatenation of VCs and a mapping between an input VCI and an output VCI performed by the ATM switches. The implication of this strategy is that dynamic bandwidth allocation is possible as applications require varying communication performance dictated by different QoS. All cells are then transferred, in sequence, over this virtual connection. ATM virtual connections may operate at either a Constant Bit Rate (CBR) or a Variable Bit Rate (VBR). ATM handles both connection-oriented and connectionless traffic. ATM also supports multiple QoS classes for differing application requirements on delay and loss performance. Thus, B-ISDN based on ATM supports a wide range of all services, such as voice, packet data (IP, etc.), video, imaging and circuit emulation.

As stated above, continuous media, such as video and voice, are very time-sensitive: the information cannot be delayed for more than a blink

of the eye, and the delay cannot have significant variations. Discrete media, such as text and image, are not nearly as delay-sensitive as continuous media. However, they are very sensitive to loss. Thus, the B-ISDN network must have the ability to discriminate between continuous media and discrete media, giving continuous media traffic priority and bounded delay, simultaneously assuring high reliability and low loss for discrete media traffic.

—Architecture Model of DMIS—

In distributed multimedia information systems, media servers and clients are connected by the high-speed network. This network is characterized by hosts (servers, clients) and switches interconnected by physical links in an arbitrary topology. The switches operate according to its corresponding design principle, introducing queuing delays, while the links introduce only propagation delays.

For DMIS, in which data applications are stored, the system has flexibility in scheduling the retrieval of data since each multimedia application service has an associated temporal presentation scenario that describes the temporal relationships among the data objects involved in the multimedia application. The structure of this scenario will be discussed in the next section. This scenario is stored in the multimedia database. Each medium may reside in its own medium server or in the database itself. The medium servers are of different types, such as image server, voice server, text and graphic server, and video server. The server has the ability to adjust the data retrieval rate of the media. Data in different media are transferred over different channels according to their traffic characteristics and transmission requirements as specified by the QoS. Through this kind of distributed multimedia information system, activities such as multimedia data retrieval and playback are performed continuously in real-time.

Multimedia Synchronization Scheme

As described in the previous section, the main cause of asynchrony in multimedia communication is jitter. Jitter consists of three parts. The first is the retrieval delay, which is the time needed for the server to collect media units and segment them into packets for transmission. The second part is the network delay from the network boundary at the server end to the boundary at the client end. The third part is the delivery delay which is the time the client needs to process the media units and prepare them for playback. Note that none of these delays are practically constant in the real world. In high-speed networks, like B-ISDN, the network delay varies because of the variable transmission rates and different queuing delays in network switches which are caused by the unpredictable traffic burstiness in the network. The retrieval and delivery delay also vary from time to time due to different processing times of media units. The processing time is dependent on the type of CPU, architecture, I/O and CPU workload of that corresponding host. Therefore, to support the transmission of time-dependent multimedia data, specific scheduling is required for storage devices, processors, and communication resources in order to overcome the latency of data, from its source to its destination.

To specify this kind of media scheduling, first we have to find a way to describe the characteristics of the media, and model the media synchronization processes in an efficient way.

—Concept Model Description of Media Synchronization—

The problem of multimedia synchronization is to satisfy temporal precedence relationships under real-time constraints. There is intra-media as well as inter-media synchronization. We characterize media synchronization and introduce a mechanism for describing this.

The problem of multimedia synchronization is to satisfy temporal precedence relationships under real-time constraints.

SIU (synchronization information unit) definition—Multimedia consists of several

medium types, such as voice, video, image, and text. For presentation of continuous media data, we specify the transmission and playout of the objects at a finer grain media rather than at the level of an object, to facilitate constant rate presentation and maintain temporal relationships among various media. The finer-grain synchronization is achieved by dividing each object into a sequence of subjects, each with its own synchronization interval, which we call synchronization information units.[19] Each SIU holds a part of the original object. The transmission of an object then consists of a stream of SIU over a channel. For the purpose of inter-stream synchronization, related objects, e.g. videoclip and audio sample that require lip-synchronization, are divided into SIUs of equal duration. Each SIU is marked with a synchronization interval number as a part of packet header information. These sequence numbers are used at the destination to determine the skew among multiple streams so that appropriate steps can be taken for realigning them. In this way, SIUs allow fine-grained synchronization. The duration of an SIU is dictated by the type of object. For instance, a video object can be divided into multiple SIUs, with each SIU holding information for a complete video frame. If an audio stream is related to a video object, both streams can be divided into SIUs having the same duration. Although the duration of an SIU for each of the two streams is the same, the size of an SIU in bytes can be different.

OCPN (object composition Petri net) model—Intra- and inter-stream synchronization schemes involved in multimedia systems are usually described by placing multimedia streams along a common time line as in digital production studios. Figure 1 shows a temporal relationship in an application consisting of a sequence of video and audio. This synchronization scheme specifies intra-synchronization constraints such as duration and relative rates of presentation of each video and audio SIU (marked as VIU and AIU, respectively). The inter-media synchronization constraints such as 'the audio must start with the video stream' and 'every two video SIUs have to be synchronized with their four related audio SIUs' can be deduced from such a description. Note that such a method does not have a well-defined inter-stream synchronization semantics. Indeed, inter-stream synchronization constraints are defined indirectly from the time line, and thus are not explicitly and clearly specified.

To specify synchronization and the relationship clearly between medium, Petri nets extended with time have been chosen for the formal modeling of multimedia process synchronization. The Timed Petri Nets model (TPN)[28] is a Petri net with time duration assigned to sites. This model allows an easy and structured specification of intra- and inter-synchronization constraints of multimedia synchronization. In fact, a medium stream is modeled as a sequence of timed sites. Therefore, using TPN for stream modeling, SIUs are modeled as timed sites. The time value associated with each site specifies its nominal duration. A more general method to specify the time relationship of multimedia objects called object composition Petri net (OCPN) is given in references 1 and 6. The OCPN has been shown to be able to capture all possible temporal relationships between objects required for multimedia presentation. The OCPN augments the conventional Petri net model with values of time, as durations, and resource utilization on the sites in the net.

DEFINITION: The OCPN is a directed graph, defined by the tuple [*T*, *P*, *A*, *M*, *D*, *R*] where:

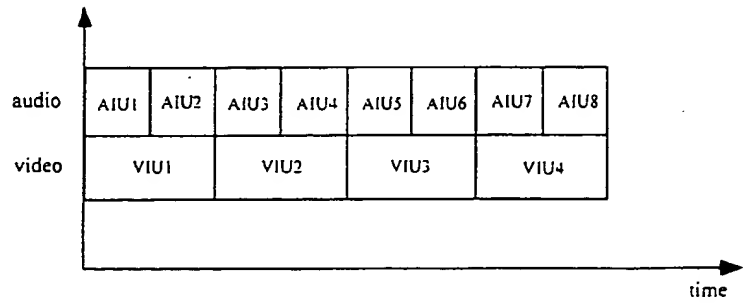

Figure 1. Temporal relationship of multimedia.

$$T = \{t_1, t_2, \ldots t_n\}$$

$$P = \{p_1, p_2, \ldots p_n\}$$

$$A: \{T \times P\} \cup \{P \times T\} \rightarrow I, I = \{1, 2, \ldots\}$$

$$M: P \rightarrow I,\ I = \{0, 1, 2, \ldots\}$$

$$D: P \rightarrow R$$

$$R: P \rightarrow \{r_1, r_2, \ldots r_n\}$$

T, P, and A represent a set of transistions (bars), a set of sites (circles), and a set of directed arcs, respectively. M assigns tokens (dots) to each place in the net. D and R are mapping from the set of sites to the real numbers (durations), and from the set of sites to a set of resources, respectively.

The firing rules governing the semantics of the OCPN model are summarized as follows:

(1) A transition t_i fires immediately when each of its input places contain an unlocked token.
(2) Upon firing, the transition t_i removes a token from each of its input places and adds a token to each of its output places.
(3) After receiving a token a site p_j remains in the active state for the interval specified by the duration t. During this interval, the token is locked. When the site becomes inactive, or upon expiration of the duration t, the token becomes unlocked.

Figures 2 and 3 are the examples of OCPN representing sequential and concurrent presentations of objects for synchronous live audio and video described in Figure 1. The OCPN template allows for characterizing live sources with representative or worst-case playout durations and object sizes.

Here, resource r_k indicates certain medium obtained from a specific media server and communication channel. Hence, the description of this corresponding data traffic can be based on this source and type, and can be associated later with specific channel delay and capacity requirements in network data transmission. With this kind of OCPN model available, it is relatively easy to choose the granularity of synchronization by assigning the time durations to the sites.

—Playout Schedule and Synchronization—

In distributed multimedia information systems, different media can start at different times and last for different durations, and they may be stored at different servers. The playout time of each medium data unit is specified so that the playout rate in terms of data units per second, determined by the nature of the medium, can be maintained. We call this a temporal playout schedule.[6] For stored data applications, every multimedia application service has the associated temporal presentation scenario that describes the temporal relationships among the media objects involved in the application. This scenario is specified exactly by the OCPN and stored in a central database.

The purpose of a synchronization mechanism is then reduced to meeting the media playout schedule specified by this presentation scenario. Both inter- and intra-media synchronization are achieved by ensuring playout of each medium and each SIU at its scheduled playout time.

Meeting playout schedule of each medium—Suppose the playout schedule for each medium of a multimedia application has been

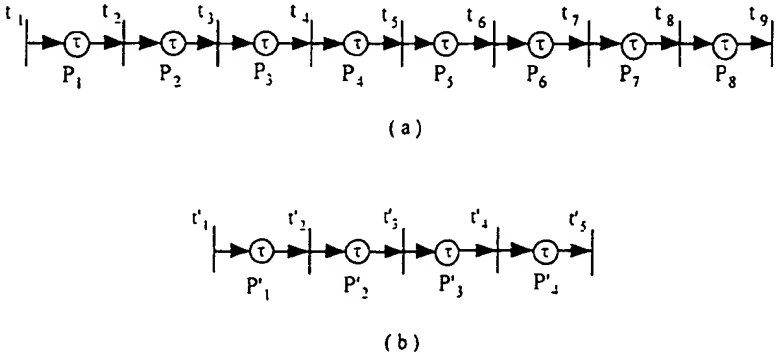

Figure 2. Modeling intra-media synchronization with OCPN.

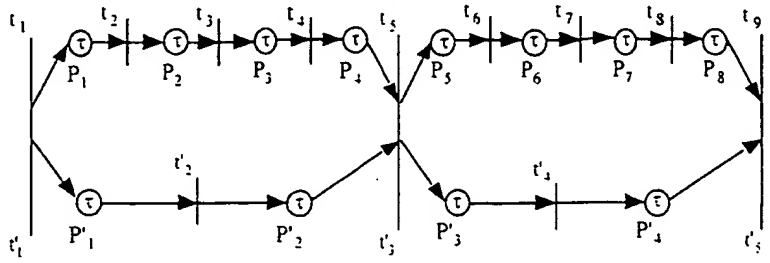

Figure 3. Modeling inter-media synchronization with OCPN.

worked out. Then media synchronization will be met if each medium unit can be presented according to the scheduled playout time.

First, let us consider a simple case where a medium stream has only one SIU with a playout time instant T. Sufficient time must be allowed to overcome the delay L caused by various processes, such as data generation, packet assembly, network access, transmission, etc. Otherwise, the deadline T will be missed. Delay L is therefore the time interval from the moment the server sends an SIU until the client receives the SIU. If we choose time, called the control time C, such that $C \geqslant L$, then the scheduled retrieval at C time units prior to T will guarantee the SIU to be presented to the client at time T and successful synchronization is guaranteed. The packet retrieval time is defined as $R = T - C$. The case of synchronization becomes the problem of meeting a single real-time deadline, which is a real-time scheduling problem. We can always find a control time C large enough to meet the deadline.[3,6] However, SIU retrieved too early will be buffered at the clients for an unnecessarily long time before it is played out. Figure 4 shows the relationship between L, C, R and T.

For a medium stream consisting of more than one SIU, since the control time takes into account the maximum delay an SIU can experience, all the data units will meet their respective playout deadlines if the medium stream is sent C time units before its scheduled time. A general scenario for synchronization of a media stream in a network is shown in Figure 5. This single medium synchronization case can be extended to multiple media streams including continuous as well as discrete media. If we send each medium at C time units prior to its scheduled playout time, it will arrive on time for presentation, and synchronization between different media streams will be maintained.

Scheduling with resource reservation— With both the playout and control time available, it is now relatively easy to carry out real-time scheduling for multimedia applications at the server side.[6,13]

First, we decompose the playout schedule specified by OCPN to each subschedule for each resource or medium r_k.

Resource decomposition algorithm:
for all places P_i in P

$$\text{if } R(P_i) = r_k \text{ then}$$
$$\Pi(r_k) = \Pi(r_k) + T_i$$
$$i(r_k) = i(r_k) + 1$$

With this subschedule available for each resource, we then reserve enough channel bandwidth for each medium. For continuous media, the packet retrieval time simply tracks the playout times with the introduced control time, that is $R_i = T_i - C_i$, for all i. The playout sequence is

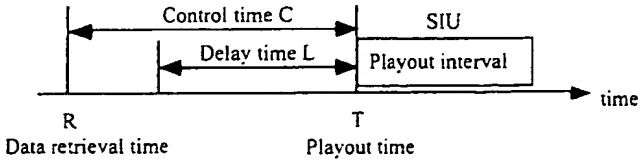

Figure 4. Definition of control time.

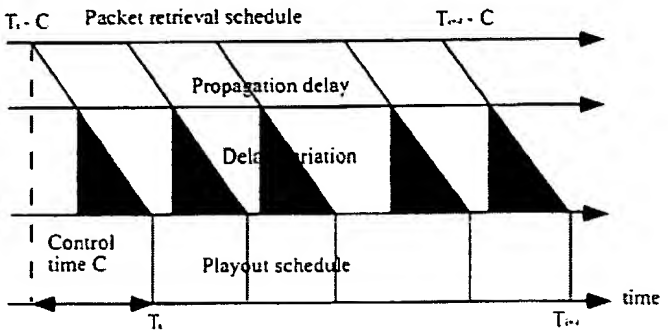

Figure 5. Synchronization of one medium stream.

merely shifted in time by C from the retrieval time. Similarly, the retrieval time for discrete media, such as image and text can be defined according to its corresponding control time, that is, $R = T$ (deadline) $- C$.

—Buffer Configuration at the Client Side—

The client receives packets of data units, created by the server and transmitted over the network, with a random delay which could be described by a delay distribution $p(t)$ (Figure 6). To reduce the delay variance of the arriving packets, a buffer has to be provided at the receiver site. It is important to note that different medium streams can have different control times and buffer requirements. This is because streams can be transferred by different channels in the network, and different media may have different data size. Moreover, data may originate from different servers at different sites. Here, we do not consider the clock drift problem, which will be discussed in the next section.

To prevent data starvation at the receiver, the estimated delays used to derive the control time are the maximum delays in the worst case. Because of the existence of network delay jitters, the actual delay experienced by each SIU may be smaller than the estimated value. Those earlier arrivals are buffered for a time duration so that each SIU will experience the same maximum delay as far as the client presentation layer is concerned. Therefore, a suitable buffer size for each medium is required at the receiver to prevent buffer overflow.[3]

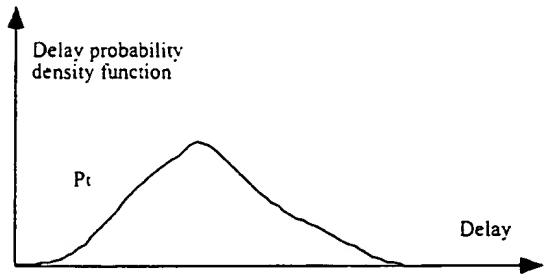

Figure 6. Delay distribution.

Size determination—Suppose we have a playout schedule for a certain medium as shown in Figure 7. Each SIU has the same duration of P units of time. The delay time of sending a SIU from a server to the arrival of this SIU to a client consists of the following (Figure 8):

(1) Time taken from the server's upper layer to fetch, assemble and send the SIU to its transport layer, which is called retrieval delay d_1.
(2) Network transmission delay from the server's transport layer to the client's transport layer, which is called transmission delay d_2.

Therefore, the first data unit is sent and arrives at the client at time $d_1 + d_2$. The second data unit will be fetched and transmitted p time units after the first SIU was fetched and transmitted, and will arrive at the client at $(d_1 + d_2 + p)$ (Figure 8). We assume d_1 and d_2 of the first SIU are statistically the same as d_1 and d_2 of the subsequent SIUs. It

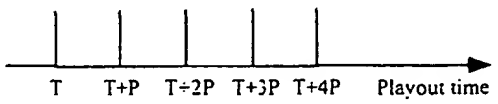

Figure 7. Playout schedule of a medium object.

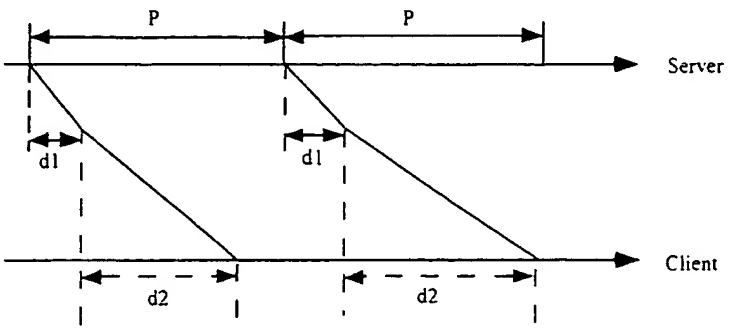

Figure 8. Illustration of delays.

can be shown that the time difference between the time a data unit is presented and the time it arrives is $C - (d_1 + d_2)$, i.e. $(d_{1max} + d_{2max}) - (d_1 + d_2)$. This time difference is the buffering time required at the receiver to make each data unit experience the same total delay before being presented. The buffering time reaches a maximum when $d_1 = d_{1min}$, $d_2 = d_{2min}$. Thus the maximum buffering time is obtained by:

$$T_b = C - (d_{1min} + d_{2min})$$

The maximum delay jitter between two SIUs is obtained by:

$$J = (d_{1max} + d_{2min})$$

The latest time a packet of a stream can leave the buffer is equal to the sum of its departure time from the server, the maximum delay from the server to the client, and the maximum buffering time T_b. In practice, if a packet has experienced maximum delay from the server to the client, it will not be buffered for the maximum buffering time T_b. Only the packet sent at the earliest departure time and experiencing minimum delay from the server to the client will be buffered for the maximum buffering time T_b. The earliest time a packet can arrive at the client is equal to the sum of its earliest departure time and the minimum delay from the server to the client. Assume the SIUs are numbered in an ascending order, 1,2,3,..., and the buffer size allocated in N data units in size. Then to prevent a buffer overflow, the SIU numbered 1 must leave the buffer before the arrival of the SIU numbered $N+1$, even in the worst case. This will be the case in which SIU 1 leaves the buffer at the latest possible time and data unit $N+1$ arrives at the earliest possible time. Therefore:

$$d_{1min} + d_{2min} + T_b \leq d_{1min} + d_{2min} + PN \qquad (1)$$

$$N \geq \frac{T_b}{P}$$

The above expression shows the minimum buffer size requirement to prevent buffer overflow is T_b/P. It is obvious that when the delays are constant and there is no delay jitter, then there is no need for buffering, i.e. $T_b = 0$. This is true because when there is no delay jitter the SIU will arrive at the client with a fixed delay and can be played out directly without buffering.

If the maximum end-to-end delay between server and client and the minimum end-to-end delay between server and client can be known in advance, then perfect synchronization can be achieved if the buffer size at the client side is configured as follows:

$$B = \left[\frac{D_{max} - D_{min}}{P}\right] \qquad (2)$$

Here $[x]$ stands for the smallest integer greater than x and P denotes the playout period of a medium at the client side. With this perfect buffer configuration at the client side, given no packet loss and no clock drift, one can always reproduce identical medium streams at the client side, and the goal of synchronization of one medium is achieved.

Intermedia synchronization—The perfect buffer size method[3] considers only one medium, in other words, only intramedia synchronization. The solution is much more complicated when con-

sidering intermedia synchronization. Whether intermedia synchronization can be obtained does not depend only on the buffer size for different media, but also on the different net delays of different media and the retrieval time at the server side. The reason for this is the size of the perfect buffer for one particular medium is related to the maximum buffering time of frame units of this medium. As indicated in the previous section, the basic idea behind the perfect buffer size is to let each frame unit experience the same total amount of delay, which is equal to the maximum delay, from the retrieval time at the server side to the presentation time at the client side. Those experiencing longer delays will be put in the buffer for a shorter time, while those experiencing the shorter delays will be put in the buffer for a longer time. Frame units which have the maximum delay will be sent directly from the buffer without staying in the buffer for any time. Frame units which have the minimum delay will stay in the buffer for the longest time, which is equal to $D_{max} - D_{min}$. The playout time for one medium is shifted from its retrieval time by the maximum delay experienced by this medium under this perfect buffer size constraint. Assuming two related media have different maximum delay, but are retrieved at the same time, then there will be contradictions on the start playout time as each medium selects its own playout time by its own maximum delay according to this perfect buffer size configuration method. The medium retrieval schedule is a critical issue here. To explain the situation in more detail, an example of this intermedia synchronization problem is given below.

Suppose there are two media, audio and video, which must be played back. The playback period is assumed to be 2. Each of these media is transmitted from separate channels according to its corresponding communication requirement. The net and operating system delays are bounded, given $D_{minv} = 1$, $D_{maxv} = 5$ for video delay and $D_{mina} = 1$, $D_{maxa} = 7$ for audio delay.

According to the perfect buffer size formula (2), a perfect buffer size for video should be 2, while for audio it should be 3. Tables 1 and 2 list the values of retrieval time, the delay experienced by video and audio, and their corresponding arrival times at the client side respectively.

Two different playout times based on the video's maximum delay and the audio's maximum delay and their corresponding buffering time for frame units are also listed in Tables 1 and 2. Column 4 in the tables lists the start playout time based on the video's maximum delay, which is 5; while column 6 lists the start playout time based on the audio's maximum delay, which is 7. If the playout time starts at 5, there will be buffer underflow at the client for audio at frame numbers 2 and 3 separately. If the playout time starts at 7, buffer overflow will happen for video at frame numbers 3, 4, and 5. To achieve intermedia synchronization with this perfect buffer size configuration, the media server must retrieve each different medium SIU by its own corresponding control time C, or the client configures the buffer size of each medium based on the maximum delay among related media if they have to be retrieved at the same time.

However, it is not easy to estimate the maximum and minimum end-to-end delays for both the operating system and network. Due to the lack of real-time operating system and real-time transport protocols,[29–32] none of these two values are bounded in the real world. Experimental results[3,32] have shown that the delay statistic varies with data unit sizes, the current traffic load of the network and the number of media streams requested simultaneously at the server. Therefore, in order to have a good estimation value of control time C, we have to analyze the characteristic of these two delay parts.

—Determination of Control Time in Stored Data Retrieval—

Based on the above description, one of the key solutions to the media synchronization problem is to find a reasonable control time for independent classes of media.[6] Clearly, control time C_i is the skew between the time an object or packet is retrieved in the server and the time played out at the client side, that is, $C_i = T_i - R_i \cdot C_i$ depends on how accurately d_1 and d_2 can be estimated, where d_1 is the retrieval delay of one SIU and d_2 is the transmission delay of one SIU. Thus, $C = d_1 + d_2$.

Estimation of control time—The retrieval control time d_1 consists of the host's processing delay, which depends heavily on the workload of the host. d_1 can be divided into two parts: d_{1min} and

Frame#	T retrieval	T delay	T arrival	T playout	T buffer	T playout	T buffer
1	0	4	4	5	1	7	3
2	2	3	5	7	2	9	4
3	4	2	6	9	3	11	5(o)
4	6	1	7	11	4	13	6(o)
5	8	2	10	13	3	15	5(o)
6	10	4	14	15	1	17	3
7	12	5	17	17	0	19	2
8	14	4	18	19	1	21	3
9	16	3	19	21	2	23	4

(o) stands for buffer overflow

Table 1. Video schedule

Frame#	T retrieval	T delay	T arrival	T playout	T buffer	T playout	T buffer
1	0	3	3	5	2	7	4
2	2	7	9	7	(u)	9	0
3	4	6	10	9	(u)	11	1
4	6	5	11	11	0	13	2
5	8	5	13	13	0	15	2
6	10	4	14	15	1	17	3
7	12	3	15	17	2	19	4
8	14	2	16	19	3	21	5
9	16	3	19	21	2	23	4

(u) stands for buffer underflow

Table 2. Audio schedule

d_{ivar}; i.e. $d_1 = d_{1min} + d_{1var}$, where d_1 is the minimum delay when there is no other workload at the server and d_{1var} is the extra delay incurred when there are other workloads. d_{1var} reaches the maximum value and d_1 becomes d_{1max} when there are maximum acceptable loads at the server.

d_2 is the end-to-end packet transmission delay incurred in the network. We can decompose this end-to-end transmission delay d_2 into three components: (1) a propagation delay D_p, which is a constant and related to the distance between the two end hosts; (2) a transmission delay D_t, which is proportional to the packet size, and is equal to P_m/C, where C is the channel capacity and P_m is the packet size for the medium m; (3) a delay variation D_v, which depends on the end-to-end network traffic. Therefore, the end-to-end transmission delay for a single packet is given by:

$$d_2 \text{ (packet)} = D_p + D_t + D_v \quad (3)$$

A typical multimedia object may consist of many packets. Let $[x]$ be the size of object x in bits. Then, $r = [x]/P_m$ gives the number of packets required to constitute the object x. The end-to-end transmission delay for an object x is then given by the following:

$$d_2 \text{ (object)} = D_p + r\,D_t + \sum_{j=1}^{r} D_{v_j} \quad (4)$$

D_v is a variable which is determined by the channel traffic and the network congestion control method. In the absence of real-time transport protocols, d_2 has to be estimated during the connection set-up phase based on the reversed resources and current network. Several approaches have been proposed to find the control time C. The network delays can be estimated either by measurement techniques or by guaranteed quality of service offered at the access to the network.[6] For

example, statistical data characterizing throughput, delay, arrival rate, interarrival rate, and buffer occupancy can be allocated using an observation mechanism.[6,32] With these collected data available, the network admission control mechanism can make resource-allocation decisions when it sets up a connection. Connections can be accepted or refused based on the available bandwidth or delay characteristics. Once accepted, the admission control guarantees that the network is able to provide the requested QoS for a connection since system congestion is prevented by the resource-allocation mechanism. Therefore, in references 6, 13 and 33. Little assumes that the underlying network transport and multiplexing mechanism provides characterization of interarrival distributions of consecutive packets through service interfaces on a per-connection basis. At the time of connection establishment, given class of traffic QoS, server and client pair, the admission control returns the channel delays and mean and variance of the interarrival time of consecutive packets. This admission control mechanism can also be extended to operating system resources including database and storage subsystems. Similarly, in reference 3 Lu suggests that with suitable admission control and resource management policies we can assume that the load on the network when the first packet is transmitted is statistically the same as the loads when subsequent packets are retrieved and transmitted. Then d_2 of the first packet will be statistically the same as d_2 of the later packets. Therefore, the meaningful control time d_2 can be established based on the maximum delay of the first packet.

—Synchronization in DMIS—

With the above mechanisms available, synchronization in DMIS becomes relatively simple. When a client requests a multimedia application, this request is first transmitted to the media database server. The database server then sends the presentation scenario related to this application back to the client. According to this scenario, the client makes connection requests to all the media servers involved in the service. This is a connection set-up phase, including request, indication, response, and confirmation. During this phase, QoS parameters specified by the application are treated as guidelines for the connection. System resources are managed to try to provide real-time data delivery. Resource reservation protocols are employed to give sufficient bandwidth and buffer space to meet the statistical characteristics of time-dependent data streams. If the required QoS cannot be met, the request for the connection is refused. The delay variations on each channel and end-to-end delays are estimated during this connection set-up, which are used to determine the buffering requirements at the client side. This buffering is used for reshaping the channel delay distribution and server load to reduce variance and to provide the requested level of service in terms of delay and late-arriving data units. If all the connections to the servers are guaranteed, the client then calculates the retrieval time of each medium according to its corresponding channel delay. We assume that the load on the servers and network during the connection established phase is statistically the same as the loads during real data transfer phase. The client sends each retrieval schedule back to its corresponding server, and the media server involved in this application starts the retrieval and transmits the media packet to the client.

In a real system there exist both short- and long-term guarantee violations in delay and bandwidth due to changes in the channel traffic characteristic or unanticipated temporary overloads in the host operating system. With the above session establishment mechanism for scheduling the retrieval time, reservation of channel resources and a suitable buffer size configuration, the media synchronization goal may still be disrupted due to buffer over- or underflow. Media packets may be lost in network transmission, i.e. via the ATM congestion control method, causing discontinuity during the playout period. Also, the perfect buffer size method gives the size of the buffer in terms of frame numbers rather than the number of bytes. In reality, it is the number of bytes, not the number of frames, that is reserved in advance during the resource reservation period. This is fine for constant frame size. With the frame numbers available, it is easy to calculate the number of bytes to be allocated first, which is equal to the frame numbers multiplied by frame size. But with the development of compression technology, almost all media frames transmitted over the net have dynamically changing sizes which depend heavily on the compression method. Yen[35] indicates the

range of frame size can be from 0.3 to 1.5 Mb. With a dynamic frame size, the number of bytes to be reserved first is usually only an estimation based on the average frame size in order to save some resources. Frame units may be lost due to buffer overflow for large frame sizes. There is a requirement to carry out some buffer adjustment for media synchronization at the client side in the presence of packet losses, temporary guarantee violations and dynamic frame sizes.

In addition, because there is no global clock in B-ISDN, the clock drift between the servers and the client side must be considered.[11,35,36] Assuming the sending rate of the server is faster than the rate of display at the client side, theoretically there will be buffer overflow at the client side. If the sending rate of the server is slower than the display rate, then the buffer will underflow at the client side. It is obvious that buffer management must be applied to control the buffer in order to overcome the clock drift problem.

Based on the above, a good synchronization mechanism consists of the following three steps. First we need to analyze the end-to-end delay, and calculate the control time and decide a suitable buffer size for different media. Second, we must schedule the retrieval time for different media based on this control time in order to eliminate the random latency. Third, a suitable buffer management must be provided at the receiver side in order to reach a fine-grain synchronization. Buffer management will be discussed in the next section.

Buffer Control System

To solve both the fine inter- and intrastream synchronization problem, one possible approach is based on controlling the client buffer. Uncontrolled buffer management will result in buffer overflow or underflow due to the network and operating sytem behavior, dynamic frame size and clock drift. Among them, delay jitter, which strongly affects the buffer behavior, is a factor which dominates the characteristic of media synchronization. Research has been done to overcome the uncertainties of the jitter property.[3,9,10,34] In this article we propose a receiver-based buffer control model that runs independently of the server. The goal is to realize the playout synchronization which it termed intrastream synchronization as well as interstream synchronization. The basic idea is to apply automatic control methodology to the buffer adjustment of the playout system.

Typically, a playback system for continuous media works in such a way that after initialization the media frames are consumed at a fixed rate out of the buffer. The buffer is assumed to compensate both the jitter effects which are introduced between frame production at the sender and frame delivery at the receiver and the error of clock drift. When there is no frame in the buffer due to continuous large network jitter, the playback system has no frame for playing out, and a glitch happens to the application users. When the buffer overflows, late-coming frames will have to be discarded. Here, we count the number of frames,[23,36] not the number of bytes, in the buffer. This is reasonable, since the frame, which has a variable size, is retrieved and played back. Assuming constant delay, server and client are synchronized when the number of frames in the buffer is held constant. However, due to network jitter effects, the arrival spacing of the frames varies temporarily, but the average spacing is constant and is determined by the server's clock. The buffer is emptied at a constant rate which is driven by the receiver's playout clock.

Our proposed approach will focus on the buffer level control on the client side. A master media concept is also introduced in order to reach intermedia synchronization. The procedure of the buffer control strategy is shown in Figure 9. The concept of nominal value[22] is treated as a reference value for frame numbers in the buffer, i.e. the frame numbers of any asynchronous media should be kept to its corresponding nominal value. The aim of the control scheme is to keep the amount of media frames in the buffer at a nominal value if possible. Skew time tolerance (Figure 10) of a human on different media will be used as the control constraints of buffer operations.

—Fundamental Definitions—

Skew tolerance—Timing parameters can characterize intermedia and real-time synchronization for the delivery of continuous as well as discrete media. For continuous media data such as audio and video, data units can be lost, which results in a gap in the playout period. Such a loss

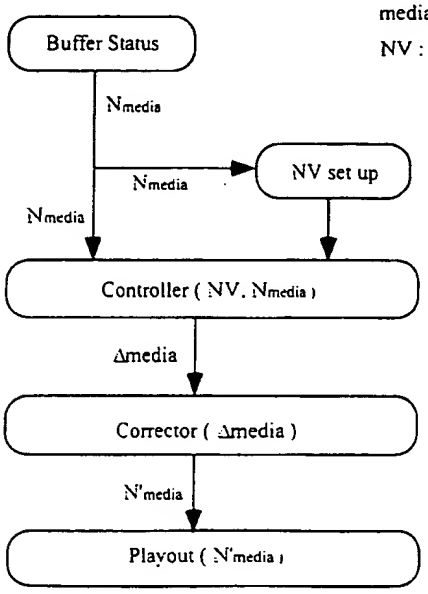

Figure 9. Flow chart of buffer control strategy.

causes the corresponding media stream to advance in time (a stream lead). Similarly, if a data unit is duplicated, it causes the stream to retard in time (a stream lag) (Figure 10). Because audio and video have substantial short-term temporal data redundancy, it is possible to control the rate of playback by dropping or duplicating data unit without affecting the quality of synchronization. The synchronization errors that can be tolerated by human perception vary in different application scenarios. Steinmetz[23] conducted some experiments in this area. For example, 120 ms mismatching in lip synchronization between an audio and video stream can be perceived as disturbance by the application users. On the other hand, with 200 ms delay for voice in a text annotation playback period, the presentation is still well accepted. These application defined parameters are called 'skew tolerance parameters' and some typical values from reference 23 are listed in Table 3.

Based on the above discussion, 'skew tolerance' could be the measurements for intermedia synchronization. Synchronization which is usually defined as an absolute instant time can be extended to a time interval, a tolerance to timing variations.[16] We specify a skew parameter sk_{ij}, which is the maximum temporal skew allowed between the ith and jth channels, given in terms of the number of durations of the playout period (1,2,...). In other words, the range of data unit values in $[X_i, Y_{j+sk_{ij}}]$ or $[X_i, Y_{j-sk_{ij}}]$ for X media on the ith channel and Y media on the jth channel is acceptable for users. For example, in digital TV application, the sk_{ij} can be {2}, i.e. the user can tolerate up to two consecutive frame index skews between the video channel and the audio channel. With the availability of this kind of skew parameter which describes the time relationship between different media, an object 1 with the playout times $P1 = \{P1_m\}$ on the ith channel is synchronized with another object 2 with the playout times $P2 = \{P2_m\}$ on the jth channel, if for all m, $|P1_m - P2_m| \leq sk_{ij}$. This skew parameter will be used later as a constraint for buffer control management to achieve intermedia as well as intramedia synchronization.

Thresholds—Basically, a buffer can compensate the normal jitter disturbance during playout if the buffer size is large enough. However, if there is a significant amount of jitter or the reserved buffer size does not match the real network delay, two extreme cases overflow and underflow will result in loss of information beyond the skew tolerance of users. Overflow happens when the number of incoming frames is greater than the remaining buffer length. Some frames to be discarded can be larger than the skew tolerance amount since

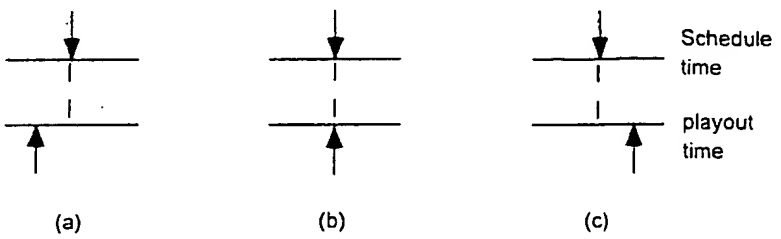

Figure 10. Skew: (a) leading, (b) none and (c) lagging.

Medium		Mode. application	QoS
Video	Animation	Correlated	±120 ms
	Audio	Lip synchronization	±80 ms
	Image	Overlay	±240 ms
		Non-overlay	±500 ms
	Text	Overlay	±240 ms
		Non-overlay	±500 ms
Audio	Animation	Event correlation (e.g. dancing)	±80 ms
	Audio	Tightly coupled (stereo)	±11 ms
		Loosely coupled (dialog mode with various participants)	±120 ms
		Loosely coupled (e.g. background music)	±500 ms
	Image	Tightly coupled (e.g. music with notes)	±5 ms
		Loosely coupled (e.g. slide show)	±500 ms
	Text	Text annotation	±240 ms
	Pointer	Audio related to showed item	±700 ms

Table 3. Skew tolerance parameters

there is no room to store the frames. The number of discarded frames is equal to the difference between the total number of current frames and the buffer length. If this number is higher than skew tolerance, inconsistent and therefore unacceptable information will be provided to the users. Underflow occurs when no new frames arrive in time due to the jitter, the media in the buffer cannot be updated, and thus a blank frame has to be played out. This is an abnormal case which seriously degrades the QoS.

To prevent the occurrence of the above case, we define three different buffer states by setting two thresholds close to both ends of the buffer (Figure 11).[33] The one near the tail of the buffer is called low threshold, while that near the head of the buffer is called high threshold. When the frame number of the media is greater than the high threshold (HT), i.e. the amount of data is close to overflow, a controller is started to reduce the difference of current frame numbers and nominal value (adjust the frame number below the high threshold HT) by discarding frames within skew tolerance. When the frame number is less than the low threshold, i.e. the amount of data in the buffer is close to the underflow status, the controller is started to reduce the difference of current frame numbers and nominal value (adjust frame number above the low threshold LT) by duplicating current playout frame within skew tolerance. Within the normal status area, the controller does not do anything. Based on these two thresholds, the number of frames which must be duplicated or discarded can be increased or decreased smoothly by duplicating or discarding gradually before under- or overflow occurs. The blank frame due to buffer underflow is replaced by duplication of the last frame in the buffer.

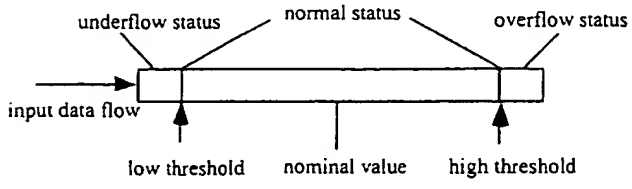

Figure 11. Three-buffer status.

The threshold plays an important role in the buffer control scheme. If the thresholds are close to the ends of the buffer, the number of duplications or discards will be reduced by increasing the risk of information loss through buffer over- and underflow. On the other hand, if the threshold is set around the middle of the buffer, there will be a higher number of frame duplication and discards since there are more opportunities to stimulate the control operational procedure. The number of buffer over- and underflows is decreased dramatically by this control procedure. The loss of information by frame duplication and discard is the price paid for the proposed buffer control scheme. The choice is between 'intentionally' lost

frame information by duplication or discards of frames via the buffer control scheme under skew constraints or 'carelessly' allowing the buffer to stay in an uncontrolled mode so that the loss of frame information depends only on delay jitter with the occurrence of buffer over- or underflow.

Theoretically, the thresholds have to be set according to the characteristic of the buffer control model and the delay jitter. The 'distance' between the thresholds and the buffer ends will reflect the adjustment time before the system reaches steady state.

Nominal value—Suppose there is a nominal value (Figure 11) to represent the number of frames which should occupy the buffer if there were no jitter and a real value of frame numbers which have actually occupied the buffer. The goal of the synchronization scheme is to keep the average buffer level (frame numbers in the buffer) to the nominal value.[23] The control model will assume the operation of adjustment by duplicating and discarding frames according to the nominal value. Since the buffer is used to compensate delay jitter, the dynamic change of buffer level is within the range of buffer compensation. The nominal value should not be fitted at one point in the buffer. Considering three different buffer statuses set by low and high thresholds as indicated above, we propose a new definition of nominal values to reach intermedia as well as intramedia synchronization. Table 4 lists various values that will be selected as nominal values according to a different buffer status for each medium. Here, N_m stands for the current frame count of one medium in its corresponding buffer. HT_m and LT_m are the thresholds in two ends of the buffer for this medium. Generally, if the frame count in the buffer is between two thresholds HT and LT, which is in a normal status, the nominal value for this medium is the current frame count number; while if the frame count of the medium is close to overflow/underflow, the nominal value for this medium is its frame count before it closes to abnormal status.

Threshold	Nominal value (NV)	Status
$N_m \geq HT_m$	$NV = HT_m - 1$	Overflow
$N_m \leq LT_m$	$NV = LT_m + 1$	Underflow
$LT_m < N_m < HT_m$	$NV = N_m$	Normal

Table 4. Nominal value for medium

Because each medium has its own medium buffer for playing out, buffer over- and underflow, frame discard or duplication happens in each separate media buffer without having any effect on other media. This results in an inconsistency of frame index between coupled medium streams. While one medium stream is playing the Nth frame, the other corresponding medium stream may play the Mth frame. The difference $(N - M)$ of the current frame index of two media should be less than the skew tolerance. With the accumulation of media discard or duplicate operations, the difference may soon be beyond the tolerance of the skew value.

A master medium-based control scheme is proposed for this intermedia synchronization problem, because a reasonable assumption for the multimedia system is that one of the media takes the role as master medium, while the others are treated as slaves relative to this master.[11,36] That means that if any frame index value of slave media is inconsistent with the frame index of its corresponding master medium, the control procedure should be used to reduce this inconsistency by adding this difference value to the current slave nominal value until both medium frame indexes are the same for playing out. Index inconsistency due to frame loss or corruption during transmission can also be dealt with in this way.

A single variable controller associated with the nominal value which is treated as a reference dominated by the buffer status and index inconsistency is proposed. The control scheme is based on the master media operational stage, i.e. any asynchronous medium should be synchronized by its corresponding nominal value or master media.

Buffer adjustment—The adjustment method of the master media-based buffer control mechanism is only processing (count, duplicate, or discard) the amount of frames. Buffer stuffing—duplication and discard of frames—are the main operations in the buffer control model.[23] When the number of frames in the buffer increases fast and close to the overflow, some will be discarded and the ratio of playout is speeded up. The frames that

should be fetched in the next time period will be played currently and the frames that should be played currently if there is no occurrence of overflow will be discarded. When the number of frames in the buffer is decreased quickly and close to underflow, the last frames will be duplicated and the frame that is fetched in the last time period will be played again. The ratio of playout is slowed down. The adjustment method is also applied to synchronize the playout index of different media based on the master media index. Slave media lagged behind the master media have to be discarded in order to 'keep pace' with the master media, while slave media, while slave media leads before the master media have to be duplicated to 'wait' for the master media. All these buffer adjustment operations (duplication or discard) must be done within the skew tolerance of different media.

Based on the above description, it is clear that with the concept of skew tolerance of users, an effective buffer control management can be applied to adjust the buffer level in order to provide graceful service degradation when buffer over- or underflow is pending or media index is inconsistent. The key idea of the proposed synchronization scheme is the effective application of a powerful automatic control approach on buffer control which will be discussed in the following sections.

—Buffer Control Scheme—

The proposed control model is composed of three parts (Figure 12): (1) comparator, (2) regulator, and (3) corrector. The comparator subtracts the frame numbers in the buffer and compares them with the nominal value, a constant which depends on different cases. The calculated deviation between the current number of frames and the nominal value is the input of the regulator which produces the corresponding output value. This output value is the number of frames which must be discarded or duplicated. According to the chosen control function, the output value of the regulator gives the amount of deviation, which has to be corrected in order to adjust the address of fetching frames to its nominal value. The corrective operation is performed by the corrector, duplicating or discarding frames by the value of this given deviation. The correction can be modeled by a proportional component with an amplification factor if it can add or discard frames without a delay. The design of the regulator is the core part in this buffer control scheme. Basically, it is a discrete state feedback controller mode from the control point of view.

Control concept—In automation engineering, control is a scheme (or an algorithm) that makes something behave according to our requirements.[37] There is open-loop as well as closed-loop control Closed-loop control, or feedback control, operates according to the following principles:

(1) Measure the variable to be controlled. Compare this measured variable with the desired value (nominal value in our case) and determine the difference.
(2) Use this difference (amplified if necessary) to adjust the controlled variable so as to reduce the difference.

Open loop is much simpler than closed loop, but it is not accurate. Usually, automatic controls are used to maintain the controlled variable at a fixed steady-state value or a changed desired value. Systems are subject to disturbances that perturb the controlled variable. With feedback control, the effect of these disturbances can be reduced more accurately. Also, by appropriate design, the response time of feedback control can be reduced substantially below that of an equivalent open-loop control. The feedback control is always unity feedback. One serious problem for feedback system is the stability: the system readily becomes an oscillator. The key issue in control system design procedure is to provide stability as well as the desired operating conditions.

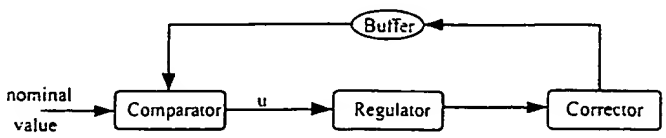

Figure 12. Buffer Control Scheme.

To develop a feedback control for a practical system we have to analyze the system. System transfer function, which is given by the Laplace transform, best measures the cause- and -effect relationship between input and output signal. A single feedback linear control system represented by a transfer function block diagram is shown in Figure 13. With this model the accuracy of the system in steady state can be studied. The equation of error, which provides the most direct measure of accuracy, is given by:

$$E(s) = \frac{U(S)}{1+G(S)} \tag{5}$$

Using the final value theorem of the Laplace transform:

$$\lim_{t\to\infty} f(t) = \lim_{S\to 0} SF(S) \tag{6}$$

We obtain

$$\lim_{t\to\infty} e(t) = \lim_{S\to 0} SE(S) = E_{steady\_state} \tag{7}$$

With the above formula, the steady-state errors when the input signal is a step can be evaluated. Equations (5)–(7) will also be used later in further analysis of the performance of the controller.

Control model—Since there is no developed model to describe the behavior of network and operating system delay, it is hard to find an accurate model to describe the buffer behavior which reflects the impact of the network delay, operating system delay jitter and clock drift. Here, we apply the automatic control method to analyze and control buffer behavior.

Three types of control function for the regulator are studied in this research. First, a very simple control function, called switch mode, is studied. This switch output mode operates proportionally and immediately to the input without any accumulative procedure. Once the calculated difference between nominal value and real frame number is larger than or less than 1, the switch mode simply duplicates or discards the current playout frame in the buffer. Second, the first-order integrator is considered. According to the small jitter disturbance input, it is reasonable to assume that the dynamic buffer level change is a first-order delay system.[23] The reason is that when the system is running, the number of frames in the buffer changes within bounds of a few frames (much less than the buffer length). This scenario is similar to the case of small-signal response at the operating point in automatic control theory. Thus the first-order integrator is the best candidate. For sudden high-level delay jitter changes due to network or operating system overload, the more exact control model should be a second-order delay which is similar to large-signal response.[23] Hence, a second-order integrator is more likely to meet the control requirements. We analyze the property of these three control functions, then propose the control system which contains these functions. The simulation results of these proposed control schemes are given in the next section.

Let us consider the model of a buffer control system represented by Laplace transformation in Figure 14. Input $U(t)$ is the difference between nominal and current values of frame numbers. This input value reflects the impact of the network, operating system delay jitter and clock drift. Output $Y(t)$ is the frame number that should be duplicated or discarded. Based on the dynamic buffer level change and its operations, we assume and attempt a second-order and first-order integral controller and switch mode, respectively.

System analysis—For a second-order controller, $G(s) = k/(s(s+a))$, while for a first-order controller, $G(s) = k/(s+a)$. Let us start with the former.

According to equation (7), we have:

$$E_{ss} = \lim_{S\to 0} \frac{SU(S)}{1+G(s)} \tag{8}$$

For step input, $U(s) = A/S$, the steady-state error for a second-order controller is:

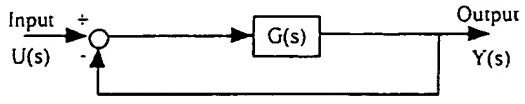

Figure 13. Block diagram of a feedback control system.

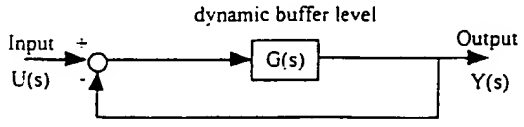

Figure 14. Block diagram of buffer controller.

$$E_{ss} = \lim_{S\to 0} \frac{S(A/S)}{1+G(s)} = \lim_{S\to 0} \frac{A}{1+G(S)} = \lim_{S\to 0} \frac{A}{\frac{K}{S(S+a)}} = 0 \quad (9)$$

Similarly, the steady-state error for a first-order controller is:

$$E_{ss} = \lim_{S\to 0} \frac{A}{1+G(S)} = \lim_{S\to 0} \frac{A}{1+\frac{K}{S+a}} = \frac{A}{1+\frac{K}{a}} = \text{constant} \quad (10)$$

For ramp input, $U(s) = A/S^2$, the steady-state error for a second-order controller is:

$$E_{ss} = \lim_{S\to 0} \frac{S(A/S^2)}{1+G(s)} = \lim_{S\to 0} \frac{A}{S+SG(S)} \quad (11)$$
$$= \lim_{S\to 0} \frac{A}{SG(S)} = \frac{aA}{K} = \text{constant}$$

The steady-state error for a first-order controller is infinite in this case.

From the above error analysis, it is clear that a first-order controller is suitable for step input, but for ramp input the error is too large. A second-order controller is suitable for both cases.

Parameter selection—The eventual goal in system design is to choose parameters, which are K and a in our case.[38] The objectives are: (1) the percent overshoot of the output to a step input $u(t)$ is less than 10% or equal to zero, (2) the steady-state error to a nominal value is minimized, and (3) the effect of the jitter disturbance is reduced.

The closed-loop output (Figure 14) is:

$$Y(s) = \frac{G(S)U(S)}{1+G(S)} \quad (12)$$

(1) *Second-order*

For a second-order system, we obtain:

$$Y(S) = \frac{k}{S^2+aS+k} U(S) \quad (13)$$

We rewrite the above equation as

$$Y(S) = \frac{\omega_n^2}{S^2+2\zeta\omega_n S+\omega_n^2} U(S) \quad (14)$$

where ζ is the dimensionless damping ratio and ω_n is the natural frequency of the system. Therefore, $K = \omega_n^2$ and $a = \zeta\omega_n$. The peak time relationship for the second-order system is

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \quad (15)$$

First, let us select k and a to meet the percent overshoot requirement for a step input, $U(s) = A/S$. We have:

$$Y(S) = \frac{k}{S^2+aS+k}\left(\frac{A}{S}\right)$$

Figure 15[38] gives the percent overshoot versus the damping ratio ζ. In order to set the overshoot at 10%, $\zeta = 0.6$ was selected according to this figure. The real overshoot will be 9.5% for this ζ according to equation (15).[38] For $\zeta = 0.9$, the overshoot will be almost zero. The steady-state error due to step disturbance is equal to zero according to the above system analysis (equation (9)). The transient response of the error due to step disturbance input can be reduced by increasing k (equation (5)).

In summary, it is better to select a large k, a proper ζ (between 0.6 and 0.9), and a large value of (k/a) in order to get a low steady-state error for the ramp input (equation (11)). For our design, K needs to be selected. Considering the characteristic equation of the system (equation (14)), we have $\omega = \sqrt{K}$ and $a = 2\zeta\omega_n$. If $\zeta = 0.9$, we have $a = 1.8\sqrt{k}$. Selecting $K = 9$, then $a = 5.4$. Realistically, K must be limited so that the system's operation remains linear.

(2)*First-order*

From the control point of view, in order for the response to be stable the real part of the roots of the characteristic equation of the system must be in the left-hand portion of the S-plane. For a first-order system, $G(s) = K/(S+a)$, so $S = -a$. The system is stable as long as $a > 0$.

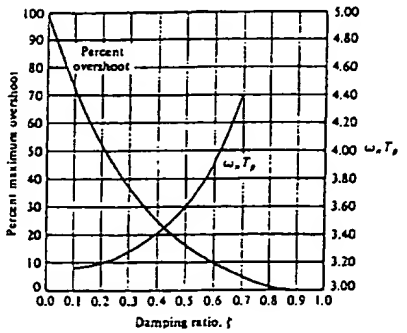

Figure 15. Percent overshoot and peak time versus damping ratio ζ for a second-order system.

If we consider step input, $U(S) = A/S$, we have

$$Y(S) = \frac{K/(S+a)}{1+K/(S+a)}\frac{A}{S} = \frac{KA}{S(S+a+K)} \quad (16)$$
$$y(t) = \frac{KA}{a+K}(1-e^{-(a+K)t})$$

From the above equation it is clear that if a or K is large, the transient response is shortened. K is a sensitive parameter to disturbance. Larger K implies greater sensitivity of measurement and more operations of the controller.

Discrete state realization—Because the real buffer operation is a discrete operation, the buffer is scanned every playout period and all calculations are done in that period. We need to describe the operation by means of the Z transform instead of Laplace transform.

Consider the Laplace representation of a second-order system:

$$G(S) = \frac{K}{S(S+a)}$$

We have

$$g(t) = \frac{K}{a}(1-e^{-at})$$
$$G(Z) = \frac{K}{a}\left(\frac{Z}{Z-1} - \frac{Z}{Z-e^{-at}}\right) = \frac{K(1-e^{-at})Z}{a(Z-1)(Z-e^{-at})}$$

Then the Z transfer function is

$$H(z) = \frac{G(Z)}{1+G(Z)} = \frac{(K/a)(1-e^{-at})Z}{Z^2+Z(K/a-1-e^{-at}-(K/a)e^{-at})+e^{-at}} \quad (17)$$

Usually, given a Z transfer function as

$$H(Z) = \frac{b_2Z^2+b_1Z+b_0}{Z^2+a_1Z+a_0} \quad (18)$$

we can obtain its discrete equation as follows:

$$y(k+2) + a_1y(k+1) + a_0y(k) = b_2u(k+2) + b_1u(k+1) + b_0u(k) \quad (19)$$

Therefore the discrete state equation for the second-order system (17) is:

$$y(k+2) + A_1y(k+1) + A_2y(k) = BU(k+1) \quad (20)$$

where

$$A_1 = \frac{K}{a} - 1 - e^{-at} - \frac{K}{a}e^{-at},\ A_2 = e^{-at}$$

and

$$B = \frac{K}{a}(1-e^{-at})$$

Let $k' = k + 2$. We have

$$y(k') = BU(k'-1) - A_1y(k'-1) - A_2y(k'-2) \quad (21)$$

For the first-order system

$$G(S) = \frac{K}{S+a}$$

therefore $g(t) = k'e^{-at}$. The corresponding Z transform is

$$G(Z) = \frac{KZ}{Z-e^{-at}}$$

Calculating the $H(Z)$ and using equations (18) and (19), the discrete state equation for the first order system is as follows,

$$y(k) = BU(k-1) - A_1y(k-1) \quad (22)$$

where $B = k/(k+1)$ and $A_1 = -e^{-at}/(k+1)$.

Here, t is the sampling period, and we assume it is the same as the playout period, a and k are the time constants which determine the performance of the control model as stated in the previous section.

—Summary of Buffer Control Scheme—

With the above buffer control scheme available, at time k, the driving input error of the control model is as follows:

$u(k)$ = frame numbers in buffer − nominal value

The output $Y(k)$ of the controller finally decides whether and how many frames need to be discarded or duplicated. The implementation of the controller can be based on three control modes: first-order, second-order, and switch. For the switch mode, the operation is simple. Once $u > 0$, $Y = -1$; once $u = 0$, $Y = 0$; once $u < 0$, $Y = 1$. For first- and second-order controller methods, K and

a need to be selected within a reasonable range in order to obtain a proper transient time response to make the controller operate more efficiently.

In summary, from the above theoretical analysis it can be seen that this control system can satisfy the requirements of buffer level control and can provide a better performance of the playout synchronization property with a skew constraint.

Simulation and Results

In this section we present simulation results to evaluate the performance of our proposed client based buffer control synchronization scheme. Because the synchronization issue includes intramedia as well as intermedia synchronization, the performance of coupled media synchronization is studied.

—Simulation Model—

Our simulation model consists of one server, one client, and the network connecting them. At the server side, a media decomposer distributes each medium stream to its corresponding medium transmitter which segments medium frames into packets and sends medium packets into one specific medium channel. At the client side, different media packets are received into their corresponding medium buffers, where medium packets are reconstructed into medium frames for playing out. The network module simulates the packet level delay characteristics of the different channels for different media. Packet delay is simulated by increasing the time stamp of an arrival packet at the medium receiver buffer by a delay value. A packet loss is simulated by not queuing the arrival packet at the medium receiver buffer. The maximum clock drift between the server and the client is assumed to be 10-4. The playback period is set to be 33 ms in the simulation. Both constant and dynamic frame sizes are studied in our simulation. Constant frame size is assumed to be 1 Mb, while dynamic frame size is assumed to be 0.5–1.5 Mb.

Network as a random delay element—For the purpose of simulation, the network is modeled as a 'black box' that introduces random delays on the packets that pass through it.[11,16,34] The model basically tries to approximate the actual delay behavior of the network in the real world. Because of the stochastic nature of the delay behavior due to a variety of reasons such as traffic flows through the network, the network switch buffer management, congestion and delay control policies adopted by the network, which are outside the scope of this article. We take the 'black box' approach[16] to simulate the network delay. Several different stochastic delay behavior models of a network have been proposed in references 5 and 11. Rangan[11] assumes an exponentially distributed network delay. Little[5] assumes a Gaussian distributed network delay. We will use these delay model assumptions (exponential and Gaussian) to study the synchronization performance of our proposed receiver based buffer control scheme in the simulation.

Buffer configuration—Besides the channel bandwidth reservation for different media during connection set-up phase, a suitable buffer size must also be chosen. According to the logic of the third section of this article, the size of the buffer is determined by

$$N \geq \frac{T_b}{P}$$

where T_b stands for the buffering time, and is equal to the difference between control time and the minimum delay in both the network and operating system, or, in other words, $T_b = C - (d_{1\min} + d_{2\min})$. P is the playout period. The perfect buffer size can be reached if we know both the maximum and minimum delays in the network and operating system, and this is described as follows:

$$B = \left[\frac{D_{\max} - D_{\min}}{P}\right]$$

Here, $[x]$ stands for the smallest integer greater than x. In reference 10 a buffer size as $[D_{\max}/P]$ is also suggested. This means packets experiencing the maximum delay are still buffered for a certain period.

It is not an easy task to obtain the maximum and minimum delays in both the network and the operating system. For the minimum delay, it is the

sum of the propagation delay and some minimum boundary delay (e.g. retrieval, transmission and delivery delay in DMIS). This relates to the topological distance, network and operating system environments between servers and clients. The maximum delay is rather arbitrary because it is related to the serving discipline of switches, the congestion control of the network and traffic in the network. We configure the buffer size as C/P based on the control time. The control time is an estimation value decided by the underlying network and operating system.

The threshold is supposed to prevent the buffer overflow or underflow status as we indicated in the previous section. Since the only operation to manage the buffer is frame duplication or discard, it is unavoidable that some frames will be lost during this control operation.

Performance measurement—The goal of the multimedia playout synchronization is to satisfy the application requirement or QoS of application, which is judged by the tolerance of a human. Thus, if the index gap between coupled media is larger than the corresponding skew value, the service will be unacceptable to users. This situation is treated as synchronization disruption. The number of disruptions is accumulated to indicate the performance of the playback system, and less is better. This criterion is the key measurement of the system performance.

Another parameter to evaluate the controlled playback system is the integrity of media information. Each frame carries unique information. Although continuous media frames may carry redundant information themselves, especially under the skew constraints, it is desirable that fewer media frames are duplicated or discarded during the controlled playback procedure. The fewer frame duplications or discards, the more integrity of information the end user gets. To prevent buffer overflow and underflow, the price we have to pay for this is discarded and duplicated frames by the controller. The loss of information is unavoidable.

—Simulation Results—

The network and operating system delays usually change from tens to hundreds of milliseconds.[3] We tested with the control time set up to 300 ms. With this control time, if the playout period is 33 ms, the buffer size is selected to be 10, according to C/P. The low and high thresholds are set to be 2 and 8, respectively. The delay distribution is assumed to be Gaussian or exponential, with a range from 50 ms to 500 ms. A total of 100,000 frames are tested.

As we proposed in the previous section, three different client-based control schemes are studied.

(1) A switch mode client-based buffer control scheme is studied with two different scenarios: (a) an ideal situation where no clock drift occurs, (b) clock drift exists between server and client, where the server's crystal oscillator clock is $10^{(-4)}$ faster or slower than the client.
(2) The first-order delay client-based buffer control scheme is studied with the above two different scenarios.
(3) The second-order delay buffer control scheme is studied with the same two scenarios.

According to the skew tolerance values listed in Table 3, it is obvious that different applications have different skew values, such as $\pm$ 80 ms for audio and video lip synchronization and $\pm$ 120 ms for loosely coupled audio and audio synchronization or for correlated video and animation. As the playout period is assumed to be 33 ms, the first skew value corresponds to two frame durations, while the second value corresponds to three. We tested the proposed control scheme under different skew values to study performance.

First, the Gaussian distribution net delay with no clock drift existing between client and server was tested. Tables 5 and 6 show the simulation results for constant frame size under different control approaches for skew values 2 and 3 separately. Tables 7 and 8 show the results for dynamic frame sizes under the same situation. Simulation results for exponential distribution delay under the same above scenarios are listed in Tables 9–12. It is clear from these tables that the number of disruptions depends heavily on the skew tolerance. The larger the skew value, the smaller the number of disruptions. The control scheme provides much better performance than no-control approaches. Without control, there is a total of 5683 disruptions in 100,000 constant size frames, while there are 906

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5688	2599	956
	Overflow	804	31	89	217
M-media	Duplicate	0	5801	2758	0
	Underflow	889	0	13	1257
S-media	Discard	0	20207	3660	1050
	Overflow	800	176	234	238
S-media	Duplicate	0	20286	3861	268
	Underflow	885	179	117	1105
Disruption		5683	906	452	371

Table 5. Performance for constant frame size under Gaussian distribution (no clock drift)

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5932	2720	964
	Overflow	804	31	89	217
M-media	Duplicate	0	6045	2892	0
	Underflow	889	0	0	1265
S-media	Discard	0	22601	4407	1201
	Overflow	800	114	156	199
S-media	Duplicate	0	22699	4624	377
	Underflow	885	98	23	1108
Disruption		2675	310	112	42

Table 6. Performance for constant frame size under Gaussian distribution (no clock drift)

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5587	2459	1820
	Overflow	913	193	308	323
M-media	Duplicate	0	5862	2834	1980
	Underflow	997	0	16	246
S-media	Discard	0	20216	3612	1996
	Overflow	912	250	355	307
S-media	Duplicate	0	20366	3929	2198
	Underflow	997	182	122	190
Disruption		9355	906	463	862

Table 7. Performance for dynamic frame size under Gaussian distribution (no clock drift)

disruptions with a switch, 452 disruptions with a first-order integrator, and only 371 disruptions with a second-order integrator for a skew value equal to 2. If the skew value becomes 3 instead of 2, there are 2675 disruptions without a control, 310 disruptions with a switch, 112 disruptions with a first-order integrator, and only 42 disruptions with a second-order integrator. The larger the skew value, the more efficiently the control scheme works. This is consistent with theoretical analysis of the control scheme. There must have been some adjusting time to provide an asymptotical

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5792	2579	641
	Overflow	913	191	308	557
M-media	Duplicate	0	6065	2967	0
	Underflow	997	0	3	1282
S-media	Discard	0	22483	4349	895
	Overflow	912	196	295	509
S-media	Duplicate	0	22681	4699	347
	Underflow	997	80	29	1142
Disruption		3227	301	109	25

Table 8. Performance for dynamic frame size under Gaussian distribution (no clock drift)

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	9810	4657	2057
	Overflow	1868	1231	1085	1031
M-media	Duplicate	0	10213	4307	0
	Underflow	1943	901	1508	3162
S-media	Discard	0	21506	7175	2690
	Overflow	1855	1120	1016	926
S-media	Duplicate	0	20465	6598	592
	Underflow	1930	2234	1667	3098
Disruption		10476	5234	3926	4221

Table 9. Performance for constant frame size under exponential distribution (no clock drift)

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	10345	4938	2065
	Overflow	1868	1660	1397	1031
M-media	Duplicate	0	11803	5940	0
	Underflow	1943	275	468	3170
S-media	Discard	0	23465	8086	2857
	Overflow	1855	1302	1147	911
S-media	Duplicate	0	23543	8369	726
	Underflow	1930	1297	938	3116
Disruption		6226	2611	1669	1369

Table 10. Performance for constant frame size under exponential distribution (no clock drift)

approach for control to work efficiently. For a dynamic frame size, this situation also holds. A second-order integrator gives the best performance of the three control schemes, especially under a skew value 3. We note that there are a few more periods of disruption than that of constant frame size under a no-control scheme. The reason is that the dynamic frame size contributes to the case of

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	9961	4808	2397
	Overflow	2426	1026	937	925
M-media	Duplicate	0	10163	4316	0
	Underflow	2500	897	1502	3048
S-media	Discard	0	21577	7245	2493
	Overflow	2431	998	939	872
S-media	Duplicate	0	20411	6591	602
	Underflow	2505	2237	1667	3079
Disruption		16589	5181	3876	4038

Table 11. Performance for dynamic frame size under exponential distribution (no clock drift)

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	10577	5178	2175
	Overflow	2426	1377	1167	1036
M-media	Duplicate	0	11730	5950	40
	Underflow	2500	277	468	3245
S-media	Discard	0	23546	8187	2920
	Overflow	2431	1153	1037	975
S-media	Duplicate	0	23464	8362	792
	Underflow	2505	1308	936	3177
Disruption		9572	2572	1662	1333

Table 12. Performance for dynamic frame size under exponential distribution (no clock drift)

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5804	2595	601
	Overflow	920	190	303	521
M-media	Duplicate	0	6078	2980	0
	Underflow	1006	0	3	1030
S-media	Discard	0	22489	4368	830
	Overflow	909	194	289	487
S-media	Duplicate	0	22684	4714	302
	Underflow	996	83	29	1089
Disruption		3088	290	108	28

Table 13. Performance for dynamic frame size under Gaussian distribution (clock drift: 1−0.0001)

buffer overflow. Dynamic frame size does not have much effect on the buffer control scheme, as the control scheme keeps counting the number of frames in the buffer to adjust the buffer level.

For the clock drift problem, Table 13 lists the synchronization performance under Gaussian distribution when a client's crystal oscillator is faster than a server's. Table 14 lists the performance when a client's crystal oscillator is slower than a server's. The skew value is 3 for both tables. From

Type of control (skew = 3)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	5778	2582	578
	Overflow	905	193	296	498
M-media	Duplicate	0	6053	2959	0
	Underflow	989	0	2	890
S-media	Discard	0	22563	4377	786
	Overflow	910	191	296	432
S-media	Duplicate	0	22751	4739	289
	Underflow	995	85	18	879
Disruption		2936	248	47	12

Table 14. Performance for dynamic frame size under Gaussian distribution (clock drift: 1+0.0001)

these tables, it is clear that when buffer overflow and underflow are dominated by delay jitter, the clock drift is not a problem in intermedia synchronization. Buffer overflow and underflow caused by clock drift is less frequent compared to delay jitter. With the buffer control scheme applied to the playback buffer, the problem caused by clock drift can also be eliminated.

As stated earlier, buffer size is the major issue in reaching synchronization in multimedia data retrieval. The simulation results provided by the tables are all obtained in a network environment as delay value changes from 50 ms to 500 ms. If the range of delay value changes from 50 ms to 400 ms, then there are only 270 disruptions out of 100,000 constant size frames with a no-control mode when the skew is 2. If the skew value is set to 3, then there is almost no disruption with a no-control method. For a skew to equal to 2 and with two thresholds set to 2 and 9 separately for low and high thresholds, the disruption will be only 190 times for a switch mode, 53 times for first-order, and 23 times for a second-order. The number of buffer overflows and underflows are greatly decreased due to this relatively perfect buffer size configuration even without any control method. Under this situation, the control method can be simplified as duplicating only the last frame when buffer starvation occurs. This is the special case of the switch method when both thresholds are set at the ends of the buffer. There are only 20 disruptions with this new method for a constant frame size, and only two disruptions for a dynamic frame size. But if the buffer size changes to 8, which means the delay range does not quite match the buffer size, buffer overflow or underflow happens more often. Then the simple switch method of duplicating the last frame does not meet the synchronization requirement; a suitable buffer control method is still needed. Tables 15 and 16 show the simulation results for buffer sizes 10 and 8 separately when the delay range is from 50 ms to 400 ms.

The simulation results show that with the perfect buffer size configuration and a suitable buffer control method applied to the receiver's buffer, media synchronization can be achieved.

Conclusion

In this article the multimedia synchronization problem in DMIS system has been studied. The emphasis is on how to overcome network and host random delays in order to provide guaranteed media synchronization.

We have described the architecture of distributed multimedia information system. A scheme to achieve media synchronization in this sytem has been presented. The scheme allows arbitrary temporary relationships between media as specified by the corresponding Petri net model. It also allows media streams to be retrieved from different servers in the network. A playout schedule to present the multimedia information units is generated based on the time relationship specification among media. The playout schedule is ensured by retrieving each data unit's C control time before its corresponding scheduled playout time. The determination of control time is a very important

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	3832	1131	478
	Overflow	47	17	42	32
M-media	Duplicate	0	3932	1256	675
	Underflow	129	0	0	0
S-media	Discard	0	16294	1711	788
	Overflow	40	58	32	10
S-media	Duplicate	0	16403	1804	689
	Underflow	129	32	22	5
Disruption		270	190	53	23

Table 15. Performance for constant frame size under Gaussian distribution 50 ms - 400 ms (size 10)

Type of control (skew = 2)		No control	Switch	First-order integrator	Second-order integrator
M-media	Discard	0	9555	3621	1467
	Overflow	596	91	160	45
M-media	Duplicate	0	9725	3853	678
	Underflow	680	4	12	0
S-media	Discard	0	28295	5562	1923
	Overflow	602	246	197	87
S-media	Duplicate	0	28382	5588	1267
	Underflow	686	242	255	145
Disruption		2897	1750	699	321

Table 16. Performance for constant frame size under Gaussian distribution 50 ms - 400 ms (size 8)

issue. We discuss the basic components of the control time and how to estimate it. The more accurate the control time, the better the synchronization performance at the client side. If the control time is not estimated correctly, the client will experience data starvation or overflow and the synchronization will fail.

To prevent the above situation, a buffer was configured based on this control time at the client side during the connection set-up phase. The main purpose of the buffer is to let each data unit experience the same total amount of delay from server side to client side in order to overcome delay jitters. Because of the dynamic behavior of network traffic and host workload, an efficient buffer management is required at the client side. A skew value decided by user applications is introduced to allow a limited buffer level adjustment. Two thresholds are set in the buffer to allow smooth control of buffer by frame duplication or discard within the skew tolerance. We define three different states for the buffer according to the different buffer levels: normal, low threshold, and high threshold. The concept of three nominal values is introduced in the buffer control scheme according to these three buffer level states. Frame stuffing (frame duplication and discard) are basic operations in the buffer control scheme. Considering intermedia synchronization, a master media-based control scheme is also provided for aligning the frame index. Three different buffer control schemes—switch, first-order integral regulator and second-order integral regulator are investigated.

The simulation results showed synchronization performance is improved with a suitable buffer size configuration and a reasonable corresponding buffer control method. With larger skew values

and larger jitter, the more efficient second-order integrator works. For small jitter change, a first-order integrator or switch works well. The simulation result also shows that determination of a perfect buffer size is a critical issue. If the delay jitter matches the buffer size, then there is no need for a complicated buffer control scheme, just allow the playback system to duplicate the last frame for playing out when buffer underflow occurs. If the buffer size is not correctly configured, then an efficient buffer control scheme is needed. From the information integrity point of view, frame duplication and discard will still lose some information. A controlled loss of information by frame duplication or discard within tolerable skew values is preferred to a random uncontrolled loss of frames by buffer overflow and underflow.

A suitable buffer size is a critical issue here to reach synchronization. This requires a reasonable control time. For this it is desirable to have a transport protocol which can provide a guaranteed QoS, and in the design of host architectures and operating systems to meet the real-time requirements. Another direction is how to better predict buffer underflow and overflow, so that the buffer control scheme can be used in a more efficient way. This can be done by monitoring the rate of queue level change in the buffer and accumulating queue statistics. Also, it is desirable that a model to describe the real delay jitter can be available, so the dynamic buffer level behavior can be deduced and a suitable control method can be designed. The setting of thresholds and their relationship with delay jitter and different media also requires more study. Delay jitter affects control model and parameter settings. Different media have different quality of service requirements. A video stream can have ample temporal redundancy, while an audio stream is more perceptible when lost, but it has periods of silence. An improved control scheme may use this feature to achieve interruptions in media synchronization.

References

1. T. D. C. Little and A. Ghafoor, Synchronization and storage models for multimedia objects, *IEEE Journal on Selected Areas in Communications*, **8**, No. 3, April, 413–27, 1990.
2. T. D. C. Little and A. Ghafoor, Network considerations for distributed multimedia object composition and communication, *IEEE Network*, November, 32–49, 1990.
3. L. H. K. Pung, T. S. Chua and S. F. Chan, Temporal synchronization support for distributed multimedia information systems, *Computer Communications*, **17**, No. 12, December, 852–62, 1994.
4. B. Furht, Multimedia systems: an overview, *IEEE Multimedia*, **1**, No. 1, Spring, 47–50, 1994.
5. R. Steinmetz, Synchronization properties in multimedia systems, *IEEE Journal on Selected Areas in Communications*, **8**, No. 3, April, 401–12, 1990.
6. T. D. C. Little and A Ghafoor, Multimedia synchronization protocols for broadband integrated services, *IEEE Journal on Selected Areas in Communications*, **9**, No. 9, December, 1368–81, 1991.
7. C. Nicolaou, An architecture for real-time multimedia communication systems,. *IEEE Journal on Selected Areas in Communications*, **8**, No. 3, April, 391–400, 1990.
8. W-H. F. Leung, T. J. Baumgartner and Y. H. Hwang, A software architecture for workstations supporting multimedia conferencing in packet switching networks, *IEEE Journal on Selected Areas in Communications*, **8**, No. 3, April, 380–89, 1990.
9. D. Ferrari, Delay jitter control scheme for packet-switching internetworks, *Computer Communications*, **15**, No. 6, July, 367–73, 1992.
10. D. Ferrari, Design and applications of a delay jitter control scheme for packet-switching internetworks, *Proc. of the 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1991, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 614, 1992, pp. 72–83.
11. V. Rangan, S. Ramanathan, H. M. Vin and T. Kaeppner, Techniques for multimedia synchronization in network file systems, *Computer Communications*, **16**, No. 3, March, 168–76, 1993.
12. V. Rangan, S. Ramanathan and T. Kaeppner, Performance of inter-media synchronization in distributed and heterogeneous multimedia systems, *Computer Networks and ISDN Systems*, **27**, 549–65, 1995.
13. T. D. C. Little and A. Ghafoor, Scheduling of bandwidth-constrained multimedia traffic, *Computer Communications*, **15**, 381–7, 1992.
14. Lamont and N. D. Georganas, Synchronization architecture and protocols for a multimedia news service application, *Proc. of the International Conference on Multimedia Computing and Systems*, pp. 3–8, 1994.
15. L. A. Karmouch and N. D. Georganas, Synchronization in real time multimedia data delivery, *Proc. IEEE International Conference on Communications*, Vol. 2, pp. 587–91, 1992.
16. S. H. Son and N. Agarwal, Synchronization of temporal constructs in distributed multimedia systems with controlled accuracy, *Proc. of the International Conference on Multimedia Computing and Systems*, pp. 550–5, 1994.
17. Ravindran and V. Bansal, Delay compensation protocols for synchronization of multimedia data

streams, *IEEE Transactions on Knowledge and Data Engineering*, 5, No. 4, August 574–89, 1993.
18. D. Shepherd and M. Salmony, Extending OSI to support synchronization required by multimedia applications, *Computer Communications*, **13**, No. 7, September, 399–406, 1990.
19. M. Woo, N. U. Qazi and A. Ghafoor, A synchronization framework for communication of pre-orchestrated multimedia information, *IEEE Network*, January/February, 52–61, 1994.
20. L. A. Karmouch and N. D. Georganas, Multimedia segment delivery scheme and its performance for real-time synchronization control, 1994 *IEEE International Conference on Communications*, Vol. 3, pp. 1734–8, 1994.
21. N. B. Pronios and T. Bozios, A scheme for multimedia and hypermedia synchronization, *Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices*, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 340–5, 1994.
22. D. E. McDysan and D. L. Spohn, *ATM: Theory and Application*, McGraw-Hill, New York, 1995.
23. D. Kohler and H. Muller, Multimedia playout synchronization using buffer level control, *Proc. of the 2nd Int. Workshop on Multimedia: Advanced Teleservices and High-speed Communication Architectures*, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 868, pp. 167–80, 1994.
24. R. Steinmetz and C. Engler, Human perception of media synchronization, IBM European networking center, Technical report, 43.9310, 1993.
25. K. Nahrstedt and R. Steinmetz, Resource management in networked multimedia systems, *Computer*, **28**, No. 5, May, 52–63, 1995.
26. L. Fedaoui, A. Seneviratne and E. Horlait, Implementation of an end-to-end Quality of Service Management scheme, *Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices*, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 124–44, 1994.
27. H. Bowman, L. Blair, G. S. Blair and A. G. Chetwynd, A formal description technique supporting expression of quality of service and media sychronization, *Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices*, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 145–67, 1994.
28. M. Diaz and P. Senac, Timed stream petri nets: a model for timed multimedia information, *Proc. of 15th International Conference on Application and Theory of Petri Nets*. 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 815, pp. 219–38, 1994.
29. D. R. C. A. Bulterman and R. Van Liere, Multimedia synchronization and UNIX, *Proc. of the 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1991, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 614, pp. 108–19, 1992.
30. T. Znati and B. Field, A network level channel abstraction for multimedia communication in real-time networks, *IEEE Transactions on Knowledge and Data Engineering*, 5, No. 4, August, 590–99, 1993.
31. B. L. Dairaine, L. Fedaoui, W. Tawbi and K. Thai, Towards an architecture for distributed multimedia applications support, *Proc. of the International Conference on Multimedia Computing and Systems*, pp. 164–72, 1994.
32. R. Steinmetz, Analyzing the multimedia operating system, *IEEE Multimedia*, **2**, No. 1, Spring, 68–84, 1995.
33. D. L. Stone and K. Jeffay, Queue monitoring: a delay jitter management policy, *4th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1993, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 846, pp. 149–60, 1994.
34. T. D. C. Little and F. Kao, An intermedia skew control system for multimedia dat presentation, *Proc. of the 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1992, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 712, pp. 130–41, 1993.
35. W. Yen and I. F. Akyildiz, On the synchronization mechansims for multimedia integrated services networks, *Proc. of International COST 237 Workshop on Multimedia Transport and Teleservices*, 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 882, pp. 168–84, 1994.
36. Z. Y. Shae, P. C. Chang and M. Chen, Capture and playback synchronization in video conferencing, *Proc. on Multimedia Computing and Networking*, 1995. SPIE, Vol. 2417, pp. 90–101, 1995.
37. G. J. Thaler, *Automatic Control Systems*, West Publishing, St Paul, MN, 1989.
38. R. C. Dorf, *Modern Control Systems*, Addison-Wesley, Reading, MA, 1992. ■

Citation

Applications, Technologies, Architectures,and Protocols for Computer Communication >archive

Proceedings of the ACM workshop on Frontiers in computer communications technology >toc

1988 , Stowe, Vermont, United States

Laboratory for emulation and study of integrated and coordinated media communication

> Also published in ...

Authors
L. F. Ludwig Bell Communications Research, Red Bank, NJ
D. F. Dunn Bell Communications Research, Red Bank, NJ

Sponsor
SIGCOMM : ACM Special Interest Group on Data Communication

Publisher
ACM Press New York, NY, USA

Pages: 283 - 291 Series-Proceeding-Article
Year of Publication: 1987
ISBN:0-89791-245-4

doi>http://doi.acm.org/10.1145/55482.55512 (Use this link to Bookmark this page)

pdf 1.05 MB

↑ ABSTRACT

In future telecommunications networks, understanding the issues of user-network control, Customer Premise Equipment (CPE) technologies, services and user applications is as important as the classical network problems of channel structure, switching, and transmission. This paper discusses a Bell Communications Research facility, the Integrated Media Architecture Laboratory (IMAL), designed to flexibly emulate a wide range of current and future network and CPE environments with a focus on multiple media communications. IMAL combines off-the-shelf technologies to create an easily clonable emulation environment for studying, planning, demonstrating, and checking the feasibility of integrated media communications. The IMAL project has assembled workstations which feature speech-synthesis/sampled-audio/telephony capabilities, local 1 MIP computation capacity, and a high-resolution color display integrating text/graphics/image/video under an expanded X Window display management system. (X Windows is an emerging windowing standard to provide high performance device-independent graphics.) The workstations may be augmented as needed by local image digitizers, video cameras, and color image printers producing paper and viewgraph hardcopies. Also, the workstations are interconnected with switches permitting access to one another as well as shared databases, temporary storage, intelligence, and information processing/conversion resources. Communications services are implemented under a distributed, real-time service primitive control scheme. This multiple-media service primitives scheme employs a threaded/dataflow-type architecture to support user-defined, network-defined, and vendor-defined services while including a wealth of flexible features for the study of network architecture, protocol, network management, and billing functions.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Scan-Tek corporate presentation at ELECTRONIC IMAGING WEST, Digital Scanners and Automatic Digitizers]or Engineerin~ Applications, Session H, Anaheim, California, February 16-19, 1987.

2 Formtek corporate presentation at ELECTRONIC IMAGING WEST, Digital $cann~re and Automatic Digitizers for Engineerinf Applications, S~ssion II, Anaheim, California, February 16-19, 1987. O.LI L. Ludwig, 'A Threaded/Flow Approach to Reconfigurable Distributed Systems and Service Primitives Architectures', Sigcom 87,

3 J. Bairstow, 'Personal Workstations Redefine Desktop Computing', High Technology, Volume 7, Number 3, March 1987, pp.18-23.

4 UNIX User's Manual, USENIX Association, March 1986 edition.

5 R.W. Scheifier, J. Clettys, 'The X window System', MIT ATHENA Project publication, October 1986 revision, Cambridge Massachusettes.

6 CONRAC Division, CONRAC Corporation, Raster Graphics Handbook~ 2rid edition, New York: Van Nostrand Reinhold Company Inc., 1985.

↑ CITINGS *4*

L. F. Ludwig, A threaded/flow approach to reconfigurable distributed systems and service primitives architectures, ACM SIGCOMM Computer Communication Review, v.17 n.5,

p.306-316, Oct./Nov. 1987

Wendy E. Mackay , Glorianna Davenport, Virtual video editing in interactive multimedia applications, Communications of the ACM, v.32 n.7, p.802-810, July 1989

Harrick M. Vin , P. Venkat Rangan , Srinivas Ramanathan, Hierarchical conferencing architectures for inter-group multimedia collaboration, ACM SIGOIS Bulletin, v.12 n.2-3, p.43-54, Nov. 1991

Chip Elliot, High-quality multimedia conferencing through a long-haul packet network, Proceedings of the first ACM international conference on Multimedia, p.91-98, August 02-06, 1993, Anaheim, California, United States

↑ **INDEX TERMS**

Primary Classification:
C. Computer Systems Organization
↳ C.2 COMPUTER-COMMUNICATION NETWORKS
↳ C.2.0 General
↳ Subjects: Data communications

Additional Classification:
C. Computer Systems Organization

General Terms:
Design

ACM SIGCOMM Computer Communication Review

Volume 17 , Issue 5 Oct./Nov. 1987

Search Results

Nothing Found

Your search for [data and storage and devices<AND>((media))] did not return any results.

You can try to rerun it within the Portal.

You may revise it and try your search again below or click advanced search for more options.

data and storage and devices<AND>((media

SEARCH [**Advanced Search**] [**Search Help/Tips**]

Complete Search Help and Tips

The following characters have specialized meaning:

Special Characters	Description
, ( ) [	These characters end a text token.
= > < !	These characters end a text token because they signify the start of a field operator. (! is special: != ends a token.)
` @ \Q < { [!	These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.

Citation

Conference on Information and Knowledge Management >archive

Proceedings of the third international conference on Information and knowledge management >toc

1994 , Gaithersburg, Maryland, United States

On the storage and retrieval of continuous media data

Authors
Banu Özden
Rajeev Rastogi
Avi Silberschatz

Sponsors
SIGIR : ACM Special Interest Group on Information Retrieval
NIST : National Institue of Standards & Technology
UMBC : U of MD Baltimore County
SIGART : ACM Special Interest Group on Artificial Intelligence

Publisher
ACM Press New York, NY, USA

Pages: 322 - 328 Series-Proceeding-Article
Year of Publication: 1994
ISBN:0-89791-674-3

doi> http://doi.acm.org/10.1145/191246.191303 (Use this link to Bookmark this page)

> full text > abstract > references > index terms > peer to peer

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format

↑ FULL TEXT: Access Rules

pdf 858 KB

↑ ABSTRACT

Continuous media applications, which require a guaranteed transfer rate of the data, are becoming an integral part of daily computational life. However, conventional file systems do not provide rate guarantees, and are therefore not suitable for the storage and retrieval of continuous media data (e.g., audio, video). To meet the demands of these new applications, continuous media file systems, which provide rate guarantees by managing critical storage resources such as memory and disks, must be designed.In this paper, we highlight the issues in the storage and retrieval of continuous media data. We first present a simple scheme for concurrently retrieving multiple continuous media streams from disks. We then introduce a a clever allocation technique for storing continuous media data that eliminates disk latency and thus, drastically reduces RAM requirements. We present, for video data, schemes for implementing the operations fast-forward, rewind and pause. Finally, we conclude by outlining directions for future research in the storage and retrieval of continuous media data.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 B. Ozdea, R. Rastogi and A. Silberschatz. 9Vellini --a file system for continuous media. Technical report, AT&T Bell Laboratories, June 1994.

2 B. (#)zden, R. Rastogi, A. Biliris and A. Silberschatz. A low-cost storage server for movie on demand databases, in Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, September 1994.

3 Clement Yu , Wei Sun , Dina Bitton , Qi Yang , Richard Bruno , John Tullis, Efficient placement of audio data on optical disks for real-time applications, Communications of the ACM, v.32 n.7, p.862-871, July 1989

4 H. a. Chen and T. D. C. Little. Physical storage organizations for time-dependent data. In Foundations of Data Organization and Algorithms, pages 19-34. Springer-Verlag, October 1993.

5 David P. Anderson , Yoshitomo Osawa , Ramesh Govindan, A file system for continuous media, ACM Transactions on Computer Systems (TOCS), v.10 n.4, p.311-337, Nov. 1992

6 R. Y. Hou, G. R. Ganger, B. L. Worthington and Y. N. Patt. Efficient storage techniques for digital continuous multimedia. IEEE Transactions orb Knowledge and Data Engineering, 5(4):564-573, August 1993.

7 Jim Gemmell , Stavros Christodoulakis, Principles of delay-sensitive multimedia data storage retrieval, ACM Transactions on Information Systems (TOIS), v.10 n.1, p.51-90, Jan. 1992

8 S. Ghandeharizadeh and L. Ramos. Continuous retrieval of multimedia data using parallelism. IEEE Transactions on Knowledge and Data Engineering, 5(4):658-669, August 1993.

9 A. Goyal, H. M. Via and P. Goyal. Algorithms for designing large-scale multimedia servers. Cornputer Communication, 1994.

10 C. L. Liu , James W. Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, Journal of the ACM (JACM), v.20 n.1, p.46-61, Jan. 1973

11 H. M. Vin, P. V. Rangan and S. Ramanathan. Designing an on-demand multimedia service. IEEE Communications Magazine, 1(1):56-64, July 1992.

12 P. V. Rangan and H. M. Vin. Efficient storage techniques for digital continuous multimedia. IEEE Transactions on Knowledge and Data Engineering, 5(4):564-573, August 1993.

13 A. L. Narasimha Reddy , James C. Wyllie, I/O issues in a multimedia system, Computer, v.27 n.3, p.69-74, March 1994

14 Chris Ruemmler , John Wilkes, An introduction to disk drive modeling, Computer, v.27 n.3, p.17-28, March 1994

↑ **INDEX TERMS**

Primary Classification:
H. Information Systems
↳ H.3 INFORMATION STORAGE AND RETRIEVAL

Additional Classification:
D. Software
↳ D.4 OPERATING SYSTEMS

H. Information Systems
↳ H.3 INFORMATION STORAGE AND RETRIEVAL

General Terms:
Theory

On the Storage and Retrieval of Continuous Media Data

Banu Özden Rajeev Rastogi Avi Silberschatz

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
{ozden, rastogi, avi}@research.att.com

Abstract

Continuous media applications, which require a guaranteed transfer rate of the data, are becoming an integral part of daily computational life. However, conventional file systems do not provide rate guarantees, and are therefore not suitable for the storage and retrieval of continuous media data (e.g., audio, video). To meet the demands of these new applications, *continuous media file systems*, which provide rate guarantees by managing critical storage resources such as memory and disks, must be designed.

In this paper, we highlight the issues in the storage and retrieval of continuous media data. We first present a simple scheme for concurrently retrieving multiple continuous media streams from disks. We then introduce a a clever allocation technique for storing continuous media data that eliminates disk latency and thus, drastically reduces RAM requirements. We present, for video data, schemes for implementing the operations fast-forward, rewind and pause. Finally, we conclude by outlining directions for future research in the storage and retrieval of continuous media data.

1 Introduction

The recent advances in compression techniques and broadband networking enable the use of continuous media applications such as multimedia electronic-mail, interactive TV, encyclopedia software, games, news, movies, on-demand tutorials, lectures, audio, video and hypermedia documents. These applications deliver to users continuous media data like video that is stored in digital form on secondary storage devices. Furthermore, continuous media-on-demand systems enable viewers to playback the media at any time and control the presentation by VCR like commands.

CIKM '94- 11/94 Gaithersburg MD USA

An important characteristic of continuous media that distinguishes it from non-continuous media (e.g., text) is that continuous media has certain timing characteristics associated with it. For example, video data is typically stored in units that are frames and must be delivered to viewers at a certain rate (which is typically 30 frames/sec). Another feature is that most continuous media types consume a large storage space and bandwidth. For example, a 100 minute movie compressed using the MPEG-I compression algorithm requires about 1.25 gigabyte (GB) of storage space. At a cost of 40 dollars per megabyte (MB), storing a movie in RAM would cost about 45,000 dollars. In comparison, the cost of storing data on disks is less than a dollar per megabyte and on tapes and CD-ROMs, it is of the order of a few cents per megabyte. Thus, it is more cost-effective to store video data on secondary storage devices like disks.

Given the limited amount of resources such as memory and disk bandwidth, it is a challenging problem to design a file system that can concurrently service a large number of both conventional and continuous media applications while providing low response time. Conventional file systems provide no rate guarantees for data retrieved and are thus unsuitable for the storage and retrieval of continuous media data. *Continuous media file systems*, on the other hand, guarantee that once a continuous media *stream* (that is, a request for the retrieval of a continuous media clip) is accepted, data for that stream is retrieved at the required rate.

The fact that the secondary storage devices have relatively high latencies and low transfer rates makes the problem more interesting. For example, besides the fact that disk bandwidths are relatively low, the disk latency imposes high buffering requirements in order to achieve a transfer rate close to the disk bandwidth. As a matter of fact, in order to support multiple streams, the closer the transfer rate gets to the disk bandwidth the higher the buffering requirements become. However, since the number of concurrent requests that can be serviced concurrently is dependent on both the buffering and bandwidth requirements, increasing transfer rate does not necessarily increase the number of requests that can be serviced concurrently.

inner track transfer rate	r_{disk}	68 Mb/sec
Settle time	t_{settle}	0.6 msec
Seek time (worst case)	t_{seek}	17 msec
Rotational latency (worst case)	t_{rot}	8.34 msec

Figure 1: The characteristics of Seagate Barracuda 2 disk.

In order to increase performance, schemes for reducing the impact of latency, as well as solutions for increasing bandwidth must be devised. Clever storage allocation schemes [2, 12, 3] as well as novel disk scheduling schemes [1, 11, 7, 13, 9] must be devised to reduce or totally eliminate latency so that buffering requirements can be reduced while bandwidth is utilized effectively. Storage techniques based on multiple disks such as replication and striping must be employed to increase the bandwidth.

In this paper, we first present a simple scheme for concurrently retrieving multiple continuous media streams from disks. We then show, how by employing novel striping techniques for storing continuous media data, we can completely eliminate disk latency and thus, drastically reduce RAM requirements. We present, for video data, schemes for implementing the basic VCR operations fast-forward, rewind, and pause. We conclude by outlining directions for future research in the storage and retrieval of continuous media data.

2 Retrieving Continuous Media Data

In this section, we briefly review characteristics of disks (additional details can be found in [14]), and present our architecture for retrieving continuous media streams from disks. We then outline a simple scheme for retrieving multiple concurrent continuous media streams from disks, and compute the buffer requirements of this scheme.

Data on disks is stored in a series of concentric circles, or *tracks*, and accessed using a disk head. A disk rotates on a central spindle and the speed of rotation denotes the transfer rate of the disk. Data on a particular track is accessed by positioning the head on (also referred to as *seeking* to) the track containing the data, and then waiting until the disk rotates enough so that the head is positioned directly above the data. Seeks typically consist of a coast during which the head moves at a constant speed and a settle, when the head position is adjusted to the desired track. Thus, the latency for accessing data on disk is the sum of seek and rotational latency. Another feature of disks is that tracks are longer at the outside than at the inside. A consequence of this is that outer tracks may have higher transfer rates than inner tracks. Figure 1 illustrates the notation we use for disk characteristics and the characteristics of the Seagate Barracuda 2 disk (we choose the disk transfer rate to be the transfer rate of the innermost track.)

In our architecture, we assume that continuous media clips are stored on disks and must be delivered at a rate r_{med}. The continuous media system is responsible for retrieving data for continuous media streams from disk into RAM at rate r_{med}. The data is then transmitted over a network to clients where they are delivered at the required rate. In this paper, we restrict ourselves to the problem at the server end – that is, the task of retrieving multiple continuous media streams from disk to RAM concurrently.

The maximum number of concurrent streams, denoted by p, that can be retrieved from disk is given by

$$p = \lfloor \frac{r_{disk}}{r_{med}} \rfloor. \quad (1)$$

A simple scheme for retrieving data for m continuous media streams concurrently is as follows. Continuous media clips are stored contiguously on disk and a buffer of size d is maintained in RAM for each of the m streams. Continuous media data is retrieved into each of the buffers at a rate r_{med} in a round robin fashion, the number of bits retrieved into a buffer during each round being d. In order to ensure that data for the m streams can be continually retrieved from disk at a rate r_{med}, in the time that the d bits from m buffers are consumed at a rate r_{med}, the d bits following the d bits consumed must be retrieved into the buffers for every one of the m streams. Since each retrieval involves positioning the disk head at the desired location and then transferring the d bits from the disk to the buffer, we have the following equation.

$$\frac{d}{r_{med}} \geq m \cdot (\frac{d}{r_{disk}} + t_{seek} + t_{rot})$$

In the above equation, $\frac{d}{r_{disk}}$ is the time it takes to transfer d bits from disk, and $t_{seek} + t_{rot}$ is the worst case disk latency. Hence, the size d of the buffer per stream can be calculated as

$$d \geq \frac{(t_{seek} + t_{rot}) \cdot r_{med} \cdot r_{disk}}{(\frac{r_{disk}}{m} - r_{med})}. \quad (2)$$

Thus, the buffer size per stream increases both with latency of the disk and the number of concurrent streams. In the following example, we compute for a commercially available disk, the buffer requirements in order to support the maximum number of concurrent streams.

Example 1: Consider MPEG-I compressed video data stored on a Seagate Baracuda 2 disk. The video data needs to be retrieved at a rate of r_{med} =1.5 Mb/s. Thus, the maximum number of streams that can be retrieved from the disk is 45. Since the worst-case rotational latency of 8.34 msec and worst case seek latency of 17 msec, the worst-case latency for the disk is 25.34 msec. From Equation 2, it follows that the minimum buffer size required in order to support 45 streams is 233 Mb. Since there is a buffer of size d for every stream, the total buffer requirements are 10 Gb. □

In the case of video streams, the VCR operations pause, fast-forward, and rewind can be implemented as follows. Pause is implemented by simply halting the consumption of bits from the buffer for the stream. Furthermore, the number of bits, d_1, read into the buffer during a round satisfies the following equality.

$$d_1 + d_2 = d$$

where d_2 is the number of unconsumed bits already contained in the buffer before data is read into it. Thus, it is possible that when a stream is paused, no data is read into the buffer for the stream until it is resumed again. Fast-forward is implemented by simply skipping a certain number of bits in the continuous media clip between the d bits retrieved during each successive round into the buffer for the stream. Similarly, rewind is implemented by retrieving preceding bits during each successive round, and skipping a certain number of bits between the d bits retrieved during successive rounds.

3 Matrix-Based Allocation

The scheme we proposed in Section 2 for retrieving data for multiple continuous media streams had high buffer requirements due to high disk latencies. In this section, we present a clever storage allocation scheme for video clips that completely eliminates disk latency and thus, keeps buffer requirements low. However, the scheme results in an increase in the worst-case response time between the time a request for a continuous media clip is made and the time the data for the stream can actually be consumed.

3.1 Storage Allocation

In order to keep the amount of buffer required low, we propose a new storage allocation scheme for continuous media clips on disk, which we call the *matrix-based* allocation scheme. This scheme is referred to as phase-constrained allocation in [2] when it is used to store a single clip. The matrix-based allocation scheme eliminates seeks to random locations, and thereby enables the concurrent retrieval of maximum number of streams p, while maintaining the buffer requirements as a constant independent of the number of streams and disk latencies. Since continuous media data is retrieved sequentially from disk, the response time for the initiation of a continuous media stream is high.

Consider a *super-clip* in which the various continuous media clips are arranged linearly one after another. Let l denote the length of the super-clip in seconds. Thus, the storage required for the super-clip is $l \cdot r_{med}$ bits. Suppose that continuous media data is read from disks in portions of size d. We shall assume that $l \cdot r_{med}$ is a multiple of $p \cdot d$.[1] Our goal is to be able to support p concurrent continuous media streams. In order to accomplish this, we divide the super-clip into

[1]The length of the super-clip can be modified by appending advertisements, etc. to the end of the super-clip.

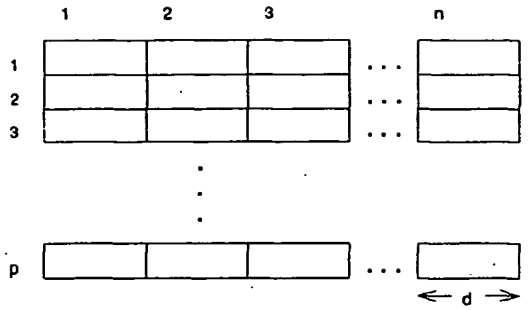

Figure 2: The super-clip viewed as a matrix.

p contiguous partitions. Thus, the super-clip can be visualized as a $(p \times 1)$ vector, the concatenation of whose rows is the super-clip itself and each row contains $t_c \cdot r_{med}$ bits of continuous media data, where

$$t_c = \frac{l}{p}.$$

Note that the first bit in any two adjacent rows are t_c seconds apart in the super-clip. Also, a continuous media clip in the super-clip may span multiple rows. Since super-clip data in each row is retrieved in portions of size d, a row can be further viewed as consisting of n portions of size d, where

$$n = \frac{t_c \cdot r_{med}}{d}$$

Thus, the super-clip can be represented as a $(p \times n)$ matrix of portions as shown in Figure 2. Each portion in the matrix can be uniquely identified by the row and column to which it belongs. Suppose we now store the super-clip matrix on disk sequentially in column-major form. Thus, as shown in Figure 3, Column 1 is stored first, followed by Column 2, and finally Column n.

We now show that by sequentially reading from disk, the super-clip data in each row can be retrieved concurrently at a rate r_{med}. From Equation 1, it follows that:

$$\frac{p \cdot d}{r_{disk}} \leq \frac{d}{r_{med}}, \tag{3}$$

Therefore, in the time required to consume d bits of continuous media data at a rate r_{med}, an entire column can be retrieved from disk. As a result, while a portion is being consumed at a rate r_{med}, the next portion can be retrieved.

Suppose that once the n^{th} column has been retrieved, the disk head can be repositioned to the start of the device almost instantaneously. In this case, we can show that p concurrent streams can be supported while the worst case response time for the initiation of a stream will be t_c. The reason for this is that every t_c seconds the disk head can be repositioned to the start. Thus, the same portion of a continuous media

clip is retrieved every t_c seconds. Furthermore, for every other concurrent stream, the last portion retrieved just before the disk head is repositioned, belongs to Column n. Since we assume that repositioning time is negligible, Column 1 can be retrieved immediately after Column n. Thus, since the portion following portion (i,n) in Column n, is portion $(i+1,1)$ in Column 1, data for concurrent streams can be retrieved from disk at a rate r_{med}. In Section 3.3, we present schemes that take into account repositioning time when retrieving data for p concurrent streams.

3.2 Buffering

We now compute the buffering requirements for our storage scheme. Unlike the scheme presented in Section 2 in which we associated a buffer with every stream, in the matrix-based scheme, with every row of the super-clip matrix, we associate a *row buffer*, into which consecutive portions in the row are retrieved. Each of the row buffers is implemented as a *circular* buffer; that is, while writing into the buffer, if the end is reached, then further bits are written at the beginning of the row buffer (similarly, while reading, if the end is reached, then subsequent bits are read from the beginning of the buffer).

With the above circular storage scheme, every $\frac{t_c}{n}$ seconds, consecutive columns of the super-clip data are retrieved from disk into row buffers. The size of each buffer is $2 \cdot d$, one half of which is used to read in a portion of the super-clip from disk, while d bits of the super-clip are consumed from the other half. Also, the number of row buffers is p. The row buffers store the p different portions of the super-clip contained in a single column – the first portion in a column is read into the first row buffer, the second portion into the second row buffer and so on. Thus, in the scheme, initially, the p portions of the super-clip in the first column are read into the first d bits of each of the corresponding row buffers. Following this, the next p portions in the second column are read into the latter d bits of each of the corresponding row buffers. Concurrently, the first d bits from each of the row buffers can be consumed for the p concurrent streams. Once the portions from the second column have been retrieved, the portions from the third column are retrieved into the first d bits of the row buffers and so on. Since consecutive portions of a super-clip are retrieved every $\frac{t_c}{n}$ seconds, consecutive portions of continuous media clips in the super-clip are retrieved into the buffer at a rate of r_{med}. Thus, in the first row buffer, the first n portions of the super-clip (from the first row) are output at a rate of r_{med}, while in the second, the next n portions (from the second row) are output and so on. As a result, a request for a continuous media stream can be initiated once the first portion of the continuous media clip is read into a row buffer. Furthermore, in the case that a continuous media clip spans multiple rows, data for the stream can be retrieved by sequentially accessing the contents of consecutive row buffers.

3.3 Repositioning

The storage technique we have presented thus far enables data to be retrieved continuously at a rate of r_{med} under the assumption that once the n^{th} column of the super-clip is retrieved from disk, the disk head can be repositioned at the start almost instantaneously. However, in practice, this assumption does not hold. Below, we present techniques for retrieving data for p concurrent streams of the super-clip if we were to relax this assumption. The basic problem is to retrieve data from the device at a rate of r_{med} in light of the fact that no data can be transferred while the head is being repositioned at the start. A simple solution to this problem is to maintain another disk which stores the super-clip exactly as stored by the first disk and which takes over the function of the disk while its head is being repositioned.

An alternate scheme, which does not require the entire super-clip to be duplicated on both disks, can be employed if t_c is at least twice the repositioning time. The super-clip data matrix is divided into two submatrices so that one submatrix contains the first $\lceil \frac{n}{2} \rceil$ columns and the other submatrix, the remaining $\lfloor \frac{n}{2} \rfloor$ columns of the original matrix, and each submatrix is stored in column-major form on two disks with bandwidth r_{disk}. The first submatrix is retrieved from the first disk, and then the second submatrix is read from the other disk while the first disk is repositioned. When the end of the data on the second disk is reached, the data is read from the first disk and the second disk is repositioned.

If the time it takes to reposition the disk to the start is low, in comparison to the time it takes to read the entire super-clip, as is the case for disks, then almost at any given instant one of the disks would be idle. To remedy this deficiency, in the following, we present a scheme that is more suitable for disks. In the scheme, we eliminate the additional disk by storing, for some m, the last m portions of the column-major form representation of the super-clip in RAM so that after the first $lr_{med} - md$ portions have been retrieved from the disk into the row buffers, repositioning of the head to the start is initiated. Furthermore, while the device is being repositioned, the last m portions of the super-clip are retrieved into the row buffers from RAM instead of the device. Once the head is repositioned and the last m portions have been retrieved into the row buffers, the columns are once again loaded into the row buffers from disk beginning with the first column as described earlier in the section. For the above scheme to retrieve data for streams at a rate of r_{med}, the time to reposition the head must be less than or equal to the time to consume m portions of continuous media data at a rate of r_{med}, that is,

$$\frac{m \cdot d}{r_{med}} \geq t_{seek} + t_{rot}$$

Thus, the total RAM required is $md + 2dp$.

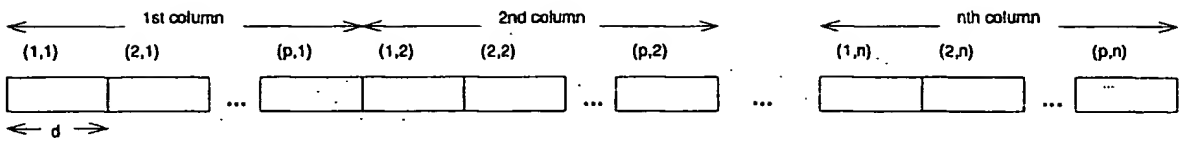

Figure 3: Placement of n columns of the super-clip matrix.

3.4 Implementation of VCR Operations

We now describe how the VCR operations begin, pause, fast-forward, rewind and resume can be implemented with the matrix-based storage architecture. As we described earlier, contiguous portions of the super-clip are retrieved into p row buffers at a rate r_{med}. The first n portions are retrieved into the first row buffer, the next n into the second row buffer, and so on.

- **begin:** The consumption of bits for a continuous media stream is initiated once a row buffer contains the first portion of the continuous media clip. Portions of size d are consumed at a rate r_{med} from the row buffer (wrapping around if necessary). After the $i \cdot n^{th}$ portion of the super-clip is consumed by a stream, consumption of data by the stream is resumed from the $i + 1^{th}$ row buffer. We refer to the row buffer that outputs the continuous media data currently being consumed by a stream as the *current* row buffer. Since in the worst case, $n \cdot d$ bits may need to be transmitted before a row buffer contains the first portion of the requested continuous media clip, the delay involved in the initiation of a stream when a begin command is issued, in the worst case, is t_c.
- **pause:** Consumption of continuous media data by the stream from the current row buffer is stopped (note however, that data is still retrieved into the row buffer as before).
- **fast-forward:** A certain number of bits are consumed from each successive row buffer following the current row buffer. Thus, during fast-forward, the number of bits skipped between consecutive bits consumed is approximately $n \cdot d$ (note that this scheme is inapplicable if successive row buffers do not contain data belonging to the same continuous media clip).
- **rewind:** This operation is implemented in a similar fashion to the fast-forward operation except that instead of jumping ahead to the following row buffer, jumps during consumption are made to the preceding row buffer. Thus, a certain number of bits are consumed from each previous row buffer preceding the current row buffer.
- **resume:** In case the previously issued command was either fast-forward or rewind, bits are continued to be consumed normally from the current row buffer. If, however, the previous command was pause, then once the current row buffer contains the bit following the last bit consumed, normal consumption of data from the row buffer is resumed beginning with the bit. Thus, in the worst case, similar to the case of the begin operation, a delay of t_c seconds may result before consumption of data for a stream can be resumed after a pause operation.

For the disk in Example 1, t_c for a 100 minute super-clip is approximately 133 seconds. Thus, the worst case delay is 133 seconds when beginning or resuming a continuous media stream. Furthermore, the number of frames skipped when fast-forwarding and rewinding is 3990 (133 seconds of video at 30 frames/s). By reducing t_c, we could reduce the worst-case response time when initiating a stream.

We now show how multiple disks can be employed to reduce t_c. Returning to Example 1, suppose that instead of using a single disk, we were to use an array of 5 disks. In this case, the bandwidth of the disk array increases from 68 Mb/s to 340 Mb/s. The number of streams, p, increases from 45 to 226, and, therefore, t_c reduces from 133 seconds to approximately 26 seconds. In this system, the worst case delay is 26 seconds and the number of frames skipped is 780 (26 seconds of video at 30 frames/sec).

4 Related Work

A number of storage schemes for continuous retrieval of video and audio data have been proposed in the literature [5, 11, 12, 7, 3, 8, 13, 9]. Among these, [5, 11, 7, 13, 9] address the problem of satisfying multiple concurrent requests for the retrieval of multimedia objects residing on a disk. These schemes are similar in spirit to the simple scheme that we presented in Section 2. In each of the schemes, concurrent requests are serviced in rounds retrieving successive portions of multimedia objects and performing multiple seeks in each round. *Admission control* tests based on computed buffer requirements for multiple requests are employed in order to determine the feasibility of additional requests with available resources. The schemes presented in [5, 11, 7] do not attempt to reduce disk latency. In [13], the authors show that the CSCAN policy for disk scheduling is superior for retrieving continuous media data in comparison to a policy in which requests with the earliest deadlines are serviced first (EDF) [10]. In [9], the authors propose a greedy disk scheduling algorithm in order to reduce both seek time and rotational latency.

In [3], in order to reduce buffer requirements, an audio record is stored on optical disk as a sequence of data blocks separated by gaps. Furthermore, in order to save disk space, the authors derive conditions for merging different audio records. In [12], similar to [3], the authors define an interleaved storage organization for multimedia data that permits the merging of time-dependent multimedia objects for efficient disk space utilization. However, they adopt a weaker condition for merging different media strands, a consequence of which is an increase in the read-ahead and buffering requirements.

In [8], the authors use parallelism in order to support the display of high resolution of video data that have high bandwidth requirements. In order to make up for the low I/O bandwidths of current disk technology, a multimedia object is declustered across several disk drives, and the aggregate bandwidth of multiple disks is utilized.

5 Research Issues

In this section, we discuss some of the research issues in the area of storage and retrieval of continuous media data that remain to be addressed.

5.1 Load Balancing and Fault Tolerance Issues

So far, we assumed that continuous media clips are stored on a single disk. However, in general, continuous media servers may have multiple disks on which continuous media clips may need to be stored. One approach to the problem is to simply partition the set of continuous media clips among the various disks and then use the schemes that we described earlier in order to store the clips on each of the disks. One problem with the approach is that if requests for continuous media clips are not distributed uniformly across the disks, then certain disks may end up idling, while others may have too much load and so some requests may not be accepted. For example, if clip C_1 is stored on disk D_1 and clip C_2 is stored on disk D_2, then if there are more requests for C_1 and fewer for C_2, then the bandwidth of disk D_2 would not be fully utilized. A solution to this problem is *striping*. By storing the first half of C_1 and C_2 on D_1 and the second half of the clips on D_2, we can ensure that the workload is evenly distributed between D_1 and D_2.

Striping continuous media clips across disks involves a number of research issues. One is the granularity of striping for the various clips. The other is that striping complicates the implementation of VCR operations. For example, consider a scenario in which every stream is paused just before data for the stream is to be retrieved from a "certain" disk D_1. If all the streams were to be resumed simultaneously, then the resumption of the last stream for which data is retrieved from D_1 may be delayed by an unacceptable amount of time. Replicating the continuous media clips across multiple disks could help in balancing the load on disks as well as reducing response times in case disks get overloaded.

Replication of the clips across disks is also useful to achieve fault-tolerance in case disks fail. One option is to use disk mirroring to recover from disk failures; another would be to use parity disks [6]. The potential problem with both of these approaches is that they are wasteful in both storage space as well as bandwidth. We need alternative schemes that effectively utilize disk bandwidth, and at the same time ensure that data for a stream can continue to be retrieved at the required rate in case of a disk failure. Finally, an interesting research issue is to vary the size of buffers allocated for streams dynamically, based on the number of streams being concurrently retrieved from disk at any point in time.

5.2 Storage Issues

Neither storing clips contiguously, nor the matrix-based storage scheme for continuous media clips is suitable in case there are frequent additions, deletions and modifications. The reason for this is that both schemes are very rigid and could lead to fragmentation. As a result, we need to consider storage schemes that decompose the storage space on disks into pages and then map various continuous media clips to a sequence of non-contiguous pages. Even though the above scheme would reduce fragmentation, since pages containing a clip may be distributed randomly across the disk, disk latency would increase resulting in increased buffer requirements. An important research issue is to determine the ideal page size for clips that would keep both space utilization high as well as disk latency low.

Another important issue to consider is the storage of continuous media clips on tertiary storage (e.g., tapes, CD-ROMs). Since continuous media data tends to be voluminous, it may be necessary (in order to reduce costs) to store it on CD-ROMs and tapes, which are much cheaper than disks. Techniques for retrieving continuous media data from tertiary storage is an interesting and challenging problem. For example, tapes have high seek times and so we may wish to use disks to cache initial portions of clips in order to keep response times low.

5.3 Data Retrieval Issues

One of the schemes we presented in the previous sections yields low response times but has significantly large buffer requirements. The other scheme, on the other hand, has much higher response times but it eliminates disk latency completely and has low buffer requirements. Further research along the lines of [1, 13, 9] must be carried out in order to reduce disk seek and rotational latency when servicing stream requests while keeping response times low.

We have also assumed so far that media streams have a single transfer rate r_{med}. This assumption may not be true. For example MPEG-I compressed video requires a transfer rate of 1.5 Mb/s, while JPEG compressed video requires a transfer rate of about 7 Mb/s [4]. We need to develop schemes to retrieve data

for continuous media streams with different transfer rates.

In the schemes we developed, we made the pessimistic assumption that the disk transfer rate r_{disk} is equal to the transfer rate of the innermost track. By taking into account the disk transfer rate of the tracks where continuous media clips are stored, we could substantially reduce buffer requirements. Also, in our work, we have not taken into account the fact that disks are not perfect. For example, disks have bad sectors that are remapped to alternate locations. Furthermore, due to thermal expansion, tables storing information on how long and how much power to apply on a particular seek, need to be recalibrated. Typically, this takes 500-800 milliseconds and occur once every 15-30 minutes. Finally, in this paper, we have only considered continuous media requests. We need a general-purpose system that would have the ability to service both continuous (e.g., video, audio) as well as non-continuous (e.g., text) media requests. Such a system would have to give a high priority to retrieving continuous media data, and use *slack* time in order to service non-continuous media requests.

6 Concluding Remarks

In this paper, we considered two approaches to retrieving continuous media data from disks. In the first approach, response times for servicing requests are low, but a high latency is incurred, resulting in significantly large buffer requirements. The second approach eliminates random disk head seeks and thus reduces buffer requirements but may result in increased response times. We presented simple schemes for implementing pause, fast-forward and rewind operations on continuous media streams (in case the data is video data). Finally, we outlined future research issues in the storage and retrieval of continuous media data.

References

[1] B. Özden, R. Rastogi and A. Silberschatz. *Fellini —a file system for continuous media*. Technical report, AT&T Bell Laboratories, June 1994.

[2] B. Özden, R. Rastogi, A. Biliris and A. Silberschatz. A low-cost storage server for movie on demand databases. In *Proceedings of the Twentieth International Conference on Very Large Databases, Santiago*, September 1994.

[3] D. Bitton, Q. Yang, R. Bruno, C. Yu, W. Sun and J. Tullis. Efficient placement of audio data on optical disks for real-time applications. *Communications of the ACM*, 32(7):862–871, July 1989.

[4] H. J. Chen and T. D. C. Little. Physical storage organizations for time-dependent data. In *Foundations of Data Organization and Algorithms*, pages 19–34. Springer-Verlag, October 1993.

[5] Y. Osawa, D. P. Anderson and R. Govindan. A file system for continius media. *ACM Transactions on Computer Systems*, 10(4):311–337, November 1992.

[6] R. Y. Hou, G. R. Ganger, B. L. Worthington and Y. N. Patt. Efficient storage techniques for digital continuous multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, August 1993.

[7] J. Gemmell and S. Christodoulakis. Principles of delay-sensitive multimedia data storage and retreival. *ACM Transactions on Information Systems*, 10(1):51–90, January 1992.

[8] S. Ghandeharizadeh and L. Ramos. Continuous retrieval of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):658–669, August 1993.

[9] A. Goyal, H. M. Vin and P. Goyal. Algorithms for designing large-scale multimedia servers. *Computer Communication*, 1994.

[10] C. L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a har real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[11] H. M. Vin, P. V. Rangan and S. Ramanathan. Designing an on-demand multimedia service. *IEEE Communications Magazine*, 1(1):56–64, July 1992.

[12] P. V. Rangan and H. M. Vin. Efficient storage techniques for digital continuous multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, August 1993.

[13] A. L. N. Reddy and J. C. Wyllie. I/O issues in a multimedia system. *Computer*, 27(3):69–74, March 1994.

[14] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *Computer*, 27(3):17–27, March 1994.